

Kilde: <https://informatikbeux.systime.dk/?id=1063>




Konstruktion af database

I dette afsnit vil vi oprette en database helt fra starten, lige fra vi får en ide, til den er konstrueret ved brug af SQL-koder. Vi kan herefter tilgå vores database ved at åbne phpMyAdmin og se vores tabeller, indsætte data mv. Det er dog vores ønske, at tilgangen til databasen kan foregå direkte fra en hjemmeside, hvilket vi ser på i [afsnit 7.6 Databaseadgang fra hjemmeside](#).

Vi betragter et skoleadministrationssystem, hvor vi har nogle lærere, der underviser en gruppe elever, der går i forskellige klasser og har forskellige fag. Vi vil opbygge databasen, så vi efterfølgende kan søge efter forskellige ting. Eksempelvis skal vi kunne søge på en elev og se, hvilke hold eleven går på. Vi kan også søge på en lærer og undersøge, hvilke hold læreren underviser.

Vi starter med en idé om, hvilke tabeller vi får behov for, og hvilke felter der skal indgå. Som udgangspunkt har vi en idé om, at vi får brug for en elevtabel, en holdtabel og en lærertabel. Vi ved også, at der bliver behov for en primærnøgle til hver tabel, som vi medtager allerede fra starten.

Vi får følgende udgangspunkt:


ELEV	HOLD	LAERER
elevid 	holdid 	initialer 
klasse	fag	navn
elevnavn	niveau	email
gade	laererinitialer	
postnr	lokale	
bynavn		
moblnr		
email		

Figur 7.22
Elevtabel, holdtabel og lærertabel med primære nøglefelter.

Vi ser her at *elevid* og *holdid* er primære nøglefelter for vores elevtabel og holdtabel, mens vi lader lærerens initialer være det primære nøglefelt for lærertabellen.

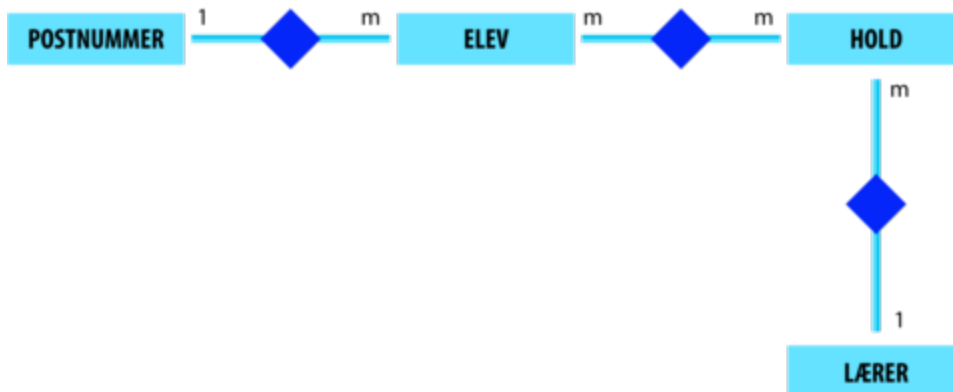
Alle tabeller har nu et nøglefelt, men elevtabellen er ikke på 3. normalform. Dette skyldes, at vi har bynavnet med i tabellen. Bynavnet afhænger direkte af, hvilket postnummer der er skrevet, og vi skal derfor oprette en ny tabel med postnumre.

Dette giver os følgende fire tabeller:

POSTNUMMER	ELEV	HOLD	LAERER
postnr 	elevnr 	holdid 	initialer 
bynavn	klasse	fag	navn
	elevnavn	niveau	email
	gade	laererinitialer	
	postnr	lokale	
	mobilnr		
	email		

Figur 7.23
Skoledatabasen med fire tabeller.

Vi kan nu konstruere et E/R-diagram med udgangspunkt i tabellerne fra figur 7.23.



Figur 7.24
Midlertidigt E/R-diagram for skoledatabasen.

I dette E/R-diagram kan vi se, at der inden for hvert postnummer kan være mange elever, mens en elev kun kan have et postnummer.

På samme vis kan en lærer undervise mange hold, men et hold har kun en lærer tilknyttet.

Mere problematisk bliver det dog med vores relation mellem elev og hold. En elev går på mange hold. På et hold er der mange elever, hvilket betyder, at vi her får en m-m relation.

Denne relation kan vi ikke arbejde videre med, når databasen og dens relationer mellem tabellerne skal oprettes elektronisk. Der er heller ikke nogle felter i de to tabeller, der kan håndtere denne relation. Vi må derfor splitte relationen op i to nye 1-m relationer og oprette en ny tabel mellem relationerne.

E/R-diagrammet kommer derefter til at se således ud:



Figur 7.25
Endeligt E/R-diagram for skoledatabasen.

Som det ses i figur 7.25, opstår der en ny tabel *holdmatch*. Der bliver derfor behov for at oprette i alt fem tabeller. Den nye tabel *holdmatch* har kun til formål at binde Elev-tabellen og Hold-tabellen sammen. Den består af tre felter. Et primært nøglefelt og to felter, der fungerer som fremmednøgler.

Vores tabeller ender med at have følgende felter:

POSTNUMMER	ELEV	HOLDMATCH	HOLD	LÆRER
postnr	elevnr	matchnr	holdid	initialer
bynavn	klasse	elevnr	fag	navn
	elevnavn	holdid	niveau	email
	gade		laererinitialer	
	postnr		lokale	
	mobilnr			
	email			

Figur 7.26
Skoledatabasens endelige struktur med fem tabeller på 3. normalform.

Efter vi har fået databasens struktur på plads, kan vi begynde at konstruere den elektronisk i MySQL.

Vi starter med at oprette databasen og derefter de fem tabeller. Det skal lige nævnes, at rækkefølgen, hvormed vi opretter vores tabeller, ikke er ligegyldig. Tabeller, der indeholder en fremmednøgle, kan først oprettes efter, at tabellen hvor deres fremmednøgle peger hen, er blevet oprettet.

Vi konstruerer databasen elektronisk ved at skrive følgende koder i MySQL:

1. Oprettelse af databasen *skoledatabase*:

```
CREATE DATABASE skoledatabase;
```

1. Oprettelse af tabellen *postnummer*:

```
CREATE TABLE postnummer (
  postnr INT(4) PRIMARY KEY,
  bynavn VARCHAR(30)
);
```

1. Oprettelse af tabellen *elev*:

```
CREATE TABLE elev (
  elevid INT AUTO_INCREMENT PRIMARY KEY,
  klasse VARCHAR(8),
  elevnavn VARCHAR(50),
  gade VARCHAR(30),
  postnr INT(4),
  mobilnr INT(8),
  email VARCHAR(50),
  FOREIGN KEY (postnr) REFERENCES postnummer (postnr)
);
```

1. Oprettelse af tabellen *laerer*:

```
CREATE TABLE laerer (
  initialer VARCHAR(4) PRIMARY KEY,
  navn VARCHAR(50),
  email VARCHAR(50)
);
```

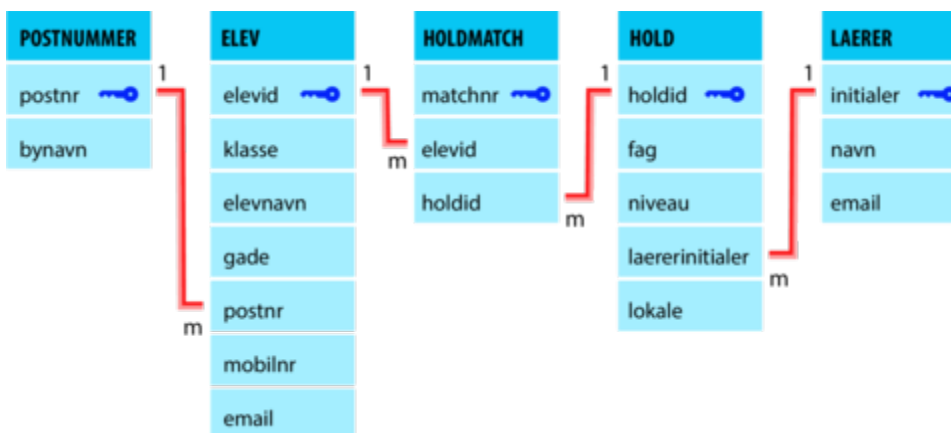
1. Oprettelse af tabellen *hold*:

```
CREATE TABLE hold (
  holdid VARCHAR(8) PRIMARY KEY,
  fag VARCHAR(20),
  niveau VARCHAR(5),
  laererinitialer VARCHAR(4),
  lokale VARCHAR(8),
  FOREIGN KEY (laererinitialer) REFERENCES laerer (initialer)
);
```

1. Oprettelse af tabellen *holdmatch*:

```
CREATE TABLE holdmatch (
  matchnr INT AUTO_INCREMENT PRIMARY KEY,
  elevid INT(10),
  holdid VARCHAR(8),
  FOREIGN KEY (elevid) REFERENCES elev (elevid),
  FOREIGN KEY (holdid) REFERENCES hold (holdid)
);
```

Ved at programmere og afvikle disse koder får vi en database, der har følgende tabeller og relationer:



Figur 7.27

Skoledatabasens endelige struktur med relationer mellem tabeller.

Herefter kan vi indsætte data i vores tabeller. Dette kan gøres direkte i vores databaseprogram, eller via en formular på en hjemmeside hvis der er oprettet forbindelse til databasen herfra.

Vi skal i dette afsnit se på et større eksempel, hvor vi fra en hjemmeside får adgang til at søge efter poster i en tabel i vores database.

Som eksempel anvender vi skoleadministrationsdatabasen, som blev oprettet i afsnit Konstruktion af database, hvor vi skal have adgang til at søge data i elevtabellen.

Eksemplet består for overskuelighedens skyld kun af koder, der vedrører funktionaliteten. Vi ser bort fra koder for design og layout.

Koderne der giver os mulighed for at søge elever i elevtabellen der bor inden for et bestemt postnummer, ser således ud:

XBI: kode kan ikke bruges, virke ikke, lav 2 PHP PDOøvelse.pdf først

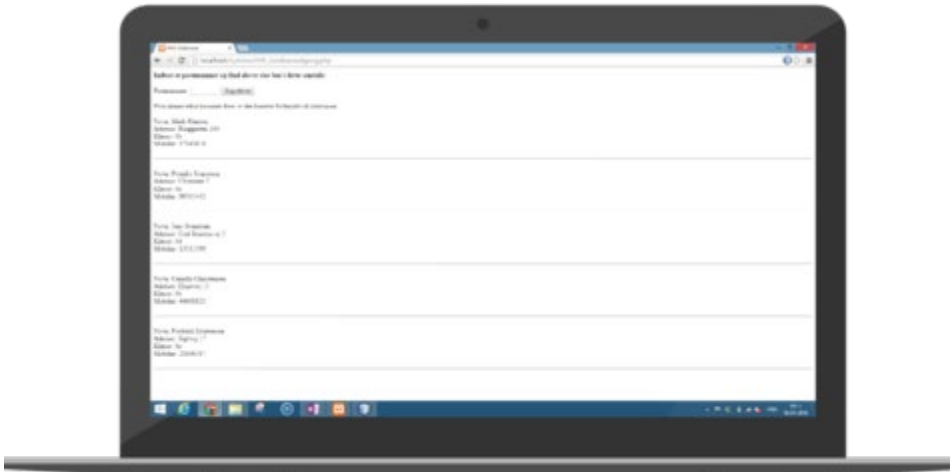
```
<!DOCTYPE html>
<html>
  <head>
    <title>PHP Database</title>
    <meta charset="UTF-8">
  </head>

  <body>
    <form action="PHP_databaseadgang.php" method="POST">
      <b>Indtast et postnummer og find elever der bor i dette område:</b><br><br>
      Postnummer: <input type="tekst" name="postnr" size="5">
      <input type="submit" value="Søg elever"/>
    </form>
    <?php
      mysql_connect("localhost" , "root" , "password");
      mysql_select_db("skoleadmin");
      echo "<br>Hvis denne tekst kommer frem er der korrekt forbundet til
databasen.<br><br>";
      $postnr = $_POST["postnr"];
      $data = mysql_query("SELECT * FROM elev WHERE (postnr) = $postnr" ) or
die(mysql_error());
      while ($row = mysql_fetch_array($data, MYSQL_ASSOC)) {
        echo "Navn: " . $row['elevnavn'] . "<br>";
        echo "Adresse: " . $row['gade'] . "<br>";
        echo "Klasse: " . $row['klasse'] . "<br>";
        echo "Mobilnr: " . $row['mobilnr'] . "<br>";
        echo "<br><hr><br>";
      }
    ?>
  </body>
</html>
```

Hovedstrukturen i disse koder består af følgende:

- Først oprettes en formular til indtastning af postnummer.

- Vælger vi at søge efter elever der bor indenfor postnummer 8270, får vi følgende skærbillede efter afvikling af vores PHP-script.



Af. xbi Opgave Uge 40

3 Upload database og php til din subdomain og test