Database design: Normalisering

Denne artikel beskriver hvor galt det kan gå, når man designer tabeller + de vigtigste værktøjer til at få orden på sit design igen: normalisering.

Artiklen fortsætter egentlig der, hvor arne_v slap med sin udemærkede artikel "Database design for begyn

Skrevet den **21. Jan 2010** af **alvion** I kategorien **Databaser / Generelt** | ★★★☆☆

Siden Eksperten fik nyt design har det været fuldstændig umuligt at se tabeleksemplerne i artiklen, hvilket gør den stort set ubrugelig. Jeg har derfor flyttet over på min egen blog, hvor jeg bedre kan kontrollere layoutet. Du kan læse den her: http://gehling.dk/2010/01/normalisering-af-databaser

Artiklen henvender sig til dig, der arbejder med databaser. metoderne beskrevet heri er ikke rettet mod bestemte database mærker. faktisk kan de også tages i anvendelse andre steder hvor du bruger datastrukturer, f.eks. kommaseparerede filer, arrays i programmer, osv.

Når du designer databaser med mange tabeller, kan du nemt komme ud for, at flere tabeller "lapper over hinanden" hvad angår de data, som de skal lagre. Det giver følgende problemer:

- Hvis indholdet af et felt i en tabel skal ændres, så skal ændringen også foretages i alle andre tabeller, hvor indholdet forekommer. Hvis dette ikke gøres, så bliver databasen pludselig *inkonsistent*.
- Hvis en tabel indeholder for mange forskellige data, kan der opstå situationer, hvor man gerne ville gemme records hvor kun en lille delmængde af felterne er udfyldt.

Nok snak for nu. Jeg vil nu vise et eksempel på, hvor galt det kunne gå. Jeg præsenterer et eksempel, som jeg så anvender igennem alle metoderne i artiklen.

Et artikelsystem

Jeg forestiller mig, at jeg har lavet et simpelt artikelsystem. Det kan pt. følgende:

- En bruger kan oprette en artikel, hvor han indtaster en overskrift, en tekst, hans eget navn og e-mail adresse.
- Andre brugere kan nu læse denne artikel og skrive kommentarer til den. En kommentar består af en dato, brugerens navn + e-mail samt kommentarteksten.

Da jeg er nybegynder i datadesign, har jeg lavet det hele i én enkelt tabel:

```
artikel
+------------+-------------------------+----------------+-----------------+--
---------------+------------------+-----------------+------------------+-----
----------------------+
| artikel_id | overskrift              | tekst          | forfatter_navn  |
forfatter_email  | kommentar_datoer | kommentar_navne | kommentar_emails |
kommentar_tekster         |
+------------+-------------------------+----------------+-----------------+--
---------------+------------------+-----------------+------------------+-----
```

```
----------------------+
|          1 | Artikel om normalisering | Bla bla bla... | Carsten Gehling |
carsten@sarum.dk | 2003-02-10       | Søren Hansen    | sh@email.dk      |
Rigtig god artikel...     |
|            |                          |                |                 |
               | 2003-02-11       | Erik Clausen    | erik@clausen.dk  | Det
var dog det arg..     |
+------------+--------------------------+----------------+-----------------+--
---------------+------------------+-----------------+------------------+-----
----------------------+
|          2 | Python for nybegyndere   | Nu skal i .... | Carsten Gehling |
carsten@sarum.dk | 2003-03-05       | Tom Andersen    | tom.and@mail.dk  | Jeg
forstår ikke h...     |
|            |                          |                |                 |
               | 2003-03-07       | Erik Clausen    | erik@clausen.dk  | Du
har endnu engang..     |
+------------+--------------------------+----------------+-----------------+--
---------------+------------------+-----------------+------------------+-----
----------------------+
|          3 | SQL injektion            | Jeg vil kort.. | Squash Guy      |
sg@eksperten.dk  | 2003-03-06       | Tom Andersen    | tom.and@mail.dk  |
Tankevækkende!!!          |
|            |                          |                |                 |
               | 2003-03-07       | Pia Jørgensen   | pj@somewhere.com | Fin
artikel men...        |
|            |                          |                |                 |
               | 2003-03-07       | Erik Clausen    | erik@clausen.dk  | Lige
et spørgsmål mere... |
+------------+--------------------------+----------------+-----------------+--
---------------+------------------+-----------------+------------------+-----
----------------------+
```

Pga. Ekspertens design, kan nogle af tabellerne i artiklen ikke ses i deres fulde bredde. Derfor har jeg også linket til dem. Ovenstående tabel kan ses her:
http://www.gehling.dk/normalisering.htm#0nf[/div]

Til dig der kender lidt mere til datadesign, og som sikkert allerede sidder og korser dig, kan jeg kun sige: Jeg har set konkrete eksempler á la ovenstående.

Metoderne her i artiklen arbejder udfra den tese, at data befinder sig i en bestemt *normalform*. Hver normalform fortæller noget om hvor velstrukturerede dine data er. Jo højere normalform jo bedre. Jeg vil beskrive de 4 første normalformer fra 0. op til og med 3. normalform og hvordan man kommer fra den ene til den næste. Normalformerne bygger på hinanden således at 3. normalform automatisk kræver at reglerne fra 1. og 2. normalform også er overholdt.

0. normalform (0NF)
Ovenstående eksempel befinder sig i 0. normalform (0NF). Dette er den løseste form, der ikke stiller nogle krav til opbygningen af dine data. Til gengæld kan du heller ikke gøre så meget ved data i denne form, uden at det kræver et stort arbejde.

Der er ingen regler for denne normalform, og derfor er der heller ikke meget mere at beskrive om den. Til gengæld vil det næste afsnit fortælle, hvilke problemer, der løses ved at ændre sine data, så de overholder reglerne for 1. normalform.

1. normalform (1NF)

Som tabellen "artikel" er opbygget lige nu, er det meget svært at udskille en enkelt kommentar fra en given artikel. Faktisk kan det ikke lade sig gøre med en SQL-sætning alene. Der skal programmeringsarbejde til. Først skal hele artikel-recorden indlæses. Derefter skal felterne "kommentar_datoer", "kommentar_navne", "kommentar_emails" og "kommentar_tekster" splittes op vha. programmering og gennemløbes.

Den måde som kommentarerne gemmes på i mit eksempel kaldes "repeating groups". Det betyder, at du har en gruppe data, som gentages i den samme record. Problemet med repeating groups er, at du som tidligere nævnt ikke kan identificere én enkelt af disse datagrupper entydigt.

Reglen for første 1. normalform (1NF) går i al sin enkelthed ud på, at dine data ikke må have "repeating groups". Dvs. alle data skal kunne identificeres entydigt udfra en nøgle (f.eks. primærnøglen).

Det var så reglen. Hvordan får vi så ændret vores data til at opfylde denne regel?

Metoden går ud på, at vi udskiller alle felterne i vores repeating groups i en separat tabel. Samtidig medtager vi primærnøglen fra tabellen "artikel" som en fremmednøgle. Dermed kan vores records i den nye tabel referere til de records i "artikel", som de hører til. Det ser således ud:

```
artikel
+------------+-------------------------+----------------+-----------------
+------------------+
| artikel_id | overskrift              | tekst          | forfatter_navn  |
forfatter_email  |
+------------+-------------------------+----------------+-----------------
+------------------+
|          1 | Artikel om normalisering | Bla bla bla... | Carsten Gehling |
carsten@sarum.dk |
+------------+-------------------------+----------------+-----------------
+------------------+
|          2 | Python for nybegyndere   | Nu skal i .... | Carsten Gehling |
carsten@sarum.dk |
+------------+-------------------------+----------------+-----------------
+------------------+
|          3 | SQL injektion            | Jeg vil kort.. | Squash Guy      |
sg@eksperten.dk  |
+------------+-------------------------+----------------+-----------------
+------------------+

kommentar
+------------+------------------+-----------------+------------------+------
---------------------+
| artikel_id | dato             | navn            | email            |
tekst                    |
+------------+------------------+-----------------+------------------+------
---------------------+
|          1 | 2003-02-10       | Søren Hansen    | sh@email.dk      |
Rigtig god artikel...     |
+------------+------------------+-----------------+------------------+------
---------------------+
|          1 | 2003-02-11       | Erik Clausen    | erik@clausen.dk  | Det
```

```
var dog det arg..     |
+------------+-----------------+----------------+-----------------+------
---------------------+
|          2 | 2003-03-05       | Tom Andersen    | tom.and@mail.dk  | Jeg
forstår ikke h...     |
+------------+-----------------+----------------+-----------------+------
---------------------+
|          2 | 2003-03-07       | Erik Clausen    | erik@clausen.dk  | Du
har endnu engang..     |
+------------+-----------------+----------------+-----------------+------
---------------------+
|          3 | 2003-03-06       | Tom Andersen    | tom.and@mail.dk  |
Tankevækkende!!!          |
+------------+-----------------+----------------+-----------------+------
---------------------+
|          3 | 2003-03-07       | Pia Jørgensen   | pj@somewhere.com | Fin
artikel men...        |
+------------+-----------------+----------------+-----------------+------
---------------------+
|          3 | 2003-03-07       | Erik Clausen    | erik@clausen.dk  | Lige
et spørgsmål mere... |
+------------+-----------------+----------------+-----------------+------
---------------------+
```

Se også [http://www.gehling.dk/normalisering.htm#1nf[/div]](http://www.gehling.dk/normalisering.htm#1nf[/div])

Begge tabeller opfylder nu kravene for 1NF.

Når du designer tabeller i et relationel databasesystem som MySQL, SQL Server, Access el. så vil du normalt havne i 1NF med det samme. Værktøjerne i systemerne lægger automatisk op til det. Men det er alligevel vigtigt at beskrive, hvad minimumskravene er for 1NF, ellers kan du nemt falde i.

Bemærk, at tabellen "kommentar" ikke har en simpel primærnøgle. Dette er bevidst gjort, for at belyse et andet aspekt ved datadesign: En primærnøgle må gerne være sammensat af flere felter. I dette tilfælde består primærnøglen faktisk af felterne "artikel_id", "dato", "email" og "tekst". Det virker måske grotesk, og det får vi også gjort noget i næste afsnit.

2. normalform (2NF)

2NF går ind og ser på sammensatte nøgler, dvs. nøgler der består af to eller flere felter i en tabel.

Det er som sagt helt i orden at have sammensatte nøgler. De fleste erfarne database designere vælger automatisk at undgå det, men de kan have deres nytte - et emne der ligger udenfor denne artikel.

Reglen for 2NF siger, at alle felter i en tabel skal være afhængig af *hele* nøglen. Det betyder, at et felt ikke må kunne findes med blot af felterne fra den sammensatte nøgle.

Vi ser nu på tabellerne fra før:

artikel
I denne tabel består primærnøglen alene af feltet "artikel_id". Da der ikke er tale om en sammensat nøgle, så opfylder tabellen automatisk kravene for 2NF.

kommentar
Som tidligere beskrevet, så består primærnøglen for denne tabel af felterne "artikel_id", "dato", "email" og "tekst". Så langt så godt. Men der er et problem. Faktisk kan feltet "navn" findes alene udfra feltet "email". Dvs. "navn" er ikke fuldt afhængigt af primærnøglen - du behøver ikke at kende værdien af alle felterne i primærnøglen, for at finde ud af, at forfatteren hedder "Tom Andersen". Dukal bare vide, at hans e-mail adresse er "tom.and@mail.dk".

Måden at løse dette problem ligner lidt måden fra før:
1) Lav en ny tabel (vi kalder den "bruger")
2) Indsæt de felter fra den gamle tabel, som kun er delvist afhængig af primærnøglen (dvs. "navn")
3) Indsæt de felter fra primærnøglen i den gamle tabel, som felterne fra 2) er afhængige af - disse bliver den nye tabels primærnøgle (i vores tilfælde er dette "email")

Det giver følgende resultat:

```
kommentar
+------------+------------------+------------------+----------------------
----+
| artikel_id | dato             | email            | tekst
   |
+------------+------------------+------------------+----------------------
----+
|          1 | 2003-02-10       | sh@email.dk      | Rigtig god artikel...
   |
+------------+------------------+------------------+----------------------
----+
|          1 | 2003-02-11       | erik@clausen.dk  | Det var dog det arg..
   |
+------------+------------------+------------------+----------------------
----+
|          2 | 2003-03-05       | tom.and@mail.dk  | Jeg forstår ikke h...
   |
+------------+------------------+------------------+----------------------
----+
|          2 | 2003-03-07       | erik@clausen.dk  | Du har endnu engang..
   |
+------------+------------------+------------------+----------------------
----+
|          3 | 2003-03-06       | tom.and@mail.dk  | Tankevækkende!!!
    |
+------------+------------------+------------------+----------------------
----+
|          3 | 2003-03-07       | pj@somewhere.com | Fin artikel men...
    |
+------------+------------------+------------------+----------------------
----+
|          3 | 2003-03-07       | erik@clausen.dk  | Lige et spørgsmål
mere... |
+------------+------------------+------------------+----------------------
----+

bruger
+------------------+-----------------+
```

```
| email             | navn            |
+-------------------+-----------------+
| sh@email.dk       | Søren Hansen    |
+-------------------+-----------------+
| erik@clausen.dk   | Erik Clausen    |
+-------------------+-----------------+
| tom.and@mail.dk   | Tom Andersen    |
+-------------------+-----------------+
| pj@somewhere.com  | Pia Jørgensen   |
+-------------------+-----------------+
```

Se også [http://www.gehling.dk/normalisering.htm#2nf[/div]](http://www.gehling.dk/normalisering.htm#2nf[/div])

I tabellen "kommentar" er feltet "email" nu fremmednøgle til tabellen "bruger". I tabellen "bruger" er feltet "email" gjort til primærnøglen.

Skulle "Erik Clausen" nu pludselig beslutte sig for at ændre sit efternavn, er det blevet noget lette at administrere. Det skal kun gøre ét sted frem for tre steder tidligere.

3. normalform (3NF)

3NF kræver, at et felt *kun* er afhængig af primærnøglen. Selvom alle felterne i en tabel overholder 2NF eller at primærnøglen slet ikke er sammensat, så kan der være tilfælde, hvor indholdet af et felt kan bestemmes alene udfra et andet felt i tabellen, som ikke selv er en del af primærnøglen.

Vi ser problemet i tabellen "artikel". "forfatter_email" er udelukkende afhængig af primærnøglen "artikel_id", men "forfatter_navn" kan faktisk findes både udfra "artikel_id" og "forfatter_email". Så tabellen overholder ikke reglen for 3NF. I lærebøgerne kaldes det at "forfatter_navn er *transitiv afhængig* af artikel_id". Dvs. forfatter_navn er afhængig af forfatter_email, som så igen er afhængig af artikel_id.

Dette løser vi således:

1) Lav en ny tabel
2) Indsæt de felter fra den gamle tabel, som kun er transitiv afhængig af primærnøglen (dvs. "forfatter_navn")
3) Indsæt de felter fra primærnøglen i den gamle tabel, som felterne fra 2) er afhængige af - disse bliver den nye tabels primærnøgle (i vores tilfælde er dette "email")

Nu er det jo så fikst, at vi allerede har en tabel med ovenstående egenskaber - nemlig tabellen "bruger". Så den tillader vi os at genbruge:

```
artikel
+------------+--------------------------+---------------+--------------
---+
| artikel_id | overskrift               | tekst         | email
  |
+------------+--------------------------+---------------+--------------
---+
|          1 | Artikel om normalisering | Bla bla bla... |
carsten@sarum.dk |
+------------+--------------------------+---------------+--------------
---+
```

```
|          2 | Python for nybegyndere   | Nu skal i .... |
carsten@sarum.dk |
+------------+--------------------------+----------------+--------------
---+
|          3 | SQL injektion            | Jeg vil kort.. |
sg@eksperten.dk  |
+------------+--------------------------+----------------+--------------
---+

bruger
+------------------+-----------------+
| email            | navn            |
+------------------+-----------------+
| carsten@sarum.dk | Carsten Gehling |
+------------------+-----------------+
| sg@eksperten.dk  | Squash Guy      |
+------------------+-----------------+
| + alle de øvrige brugere fra før.. |
+------------------+-----------------+
```

Se også [http://www.gehling.dk/normalisering.htm#3nf[/div]](http://www.gehling.dk/normalisering.htm#3nf[/div])

Nu overholder alle tabellerne både 1., 2. og 3. normalform. Dermed er du også kommet ud over de problemer, som jeg beskrev i starten:

- Hvis en bruger tidligere skulle skifte navn, så skulle det både ændres i alle records, hvor brugeren havde oprettet artikler og skrevet kommentarer. Nu skal navnet bare ændres én gang i tabellen bruger.
- Det var ikke muligt at oprette en bruger uden samtidig at oprette en artikel. Hvis du gjorde, så ville recorden i den gamle artikel tabel indeholde mangle tomme felter. Det er nu muligt brugeren skal bare oprettes i tabellen bruger.

Afslutning

Der kunne godt laves andre optimeringer på tabellerne. F.eks. er det en god idé at oprette et numerisk felt f.eks. kaldet "bruger_id" som primærnøgle i "bruger" og fremmednøgle i "artikel". Ellers er det sværere at opdatere en brugers e-mail. Men det er ikke et krav for 1NF, 2NF eller 3NF.

Metoden bringer adskiller data i yderste konsekvens. Det betyder også, at visse dataudtræk (SELECT) kræver, at to eller flere tabeller join'es, en operation der tager længere tid end en simpel select. Derfor kan du somme tider komme ud for, at det godt kan betale sig at bryde med en af reglerne af hensyn til performance.

MEN: Lær først reglerne grundigt. Så er du også mere kompetent til at bryde dem de rigtige steder. :-)

God fornøjelse.

Kommentar af nuna d. 23. Mar 2004 | 1

Kommentar af simonvalter d. 03. May 2004 | 2

Kommentar af jensgram d. 22. Oct 2004 | 3

Lærerigt - et særdeles godt afslutningsafsnit på en fornem artikel.

Kommentar af hiks d. 02. Nov 2004 | 4

Fin artikel for både nybegyndere og autodidakte. :o)

Kommentar af htmlaga d. 27. Mar 2004 | 5

Kommentar af phineas_phreak d. 02. Apr 2004 | 6

Kommentar af thundergod d. 29. Jun 2004 | 7

Kommentar af al1407 d. 24. Jun 2004 | 8

Meget godt for nybegyndere, eller folk der bare vil lære mere :)

Kommentar af godzs d. 21. Mar 2004 | 9

Kommentar af detox d. 31. Mar 2004 | 10

Godt og enkelt forklaret.

Kommentar af dorteboeg d. 10. Aug 2004 | 11

Kommentar af keeem d. 16. Oct 2008 | 12

Fin artikel, læs også mere om Normalisering af databaser her.

Kommentar af domaz.dk d. 23. Mar 2004 | 13

Mange gode små tips.... for dem der ikke er så meget inde i det...

Men henvender sig nok mest til "nybegyndere"...

Kommentar af kazine d. 19. Jun 2007 | 14

Kommentar af janus_007 d. 22. Jan 2010 | 15

Glem ikke Boyce Codd, manden der startede hele imperiet med relationelle baser :)

Kommentar af fjappe d. 22. Jan 2010 | 16

God artikel, forklarer let forståeligt, uden brug af alt for mange ord der ikke er til at forstå

alliggevel, hvordan man kommer igennem de 3 normalformer :)

Kommentar af kimsey0 d. 24. Jan 2010 | 17

Glimrende artikkel, men nej hvor er det altså irriterende at hele siden bliver indrammet i en stor lyseblå boks af at komme herind.
[/div]

Kommentar af showsource d. 27. Jan 2010 | 18

Rimelig artikel, men jeg er nu ikke meget for at email står i flere tabeller.
Skulle det ikke være id for det row hvor email findes ?
Altså

user:
id, autoincrement, primary
navn, navn
email, email

artikler:
artikel_id, autoincrement, primary
forfatter_id, svarerende til id i user tabel
tekst, blabla

o.s.v.