SQL og ASP

En artikel omkring simpel SQL og hvordan disse opbygges, udformes og udføres, sådan at man kan få et brugbart resultat i ASP. Dette ligefra forklaring til opbygningen af SQL'en og forbindelse til databasen.

Skrevet den **06. Feb 2009** af **hiks** I kategorien **Programmering / ASP** | ★☆☆☆☆

SQL er et forespørgselssprog, og er en forkortelse af *Structured Query Language*. Frit oversat til dansk betyder det "struktureret forespørgselssprog".

SQL bruges til at lave Forespørgsler/Querys. Disse forespørgsler kan både bruges til generering af indholdet i vores ASP-sider, men kan også bruges til at manipulere i selve databasen med. Det vil sige at SQL også kan oprette nye tabeller og foretage mange andre herlige ting til gavn for mange mennesker der professionelt arbejder med databaser. Dette er dog ikke beskrevet i denne artikel.

Et simpelt SQL-udtryk
I SQL er der en række reserverede ord. Disse ord bør for god ordens skyld altid skrives med BLOKBOGSTAVER, for at disse kan kendes fra den anden tekst i programmeringskoden. Principielt er det ligegyldigt for ASP, men hvis vi skal følge god programmeringsskik og følge standarden, så skal de reserverede ord skrives med STORT.

Listen over reserverede ord er meget lang, men de væsentligste vil jeg beskrive her i denne artikel:

Den generelle SQL-sætning er opbygget som følger:

1. SELECT "Attributnavn(e)" (eller et beregningsudtryk)
2. FROM "Tabelnavn(e)"
3. WHERE betingelser;

Når du arbejder med SQL i ASP vil du altid definere disse - og de kan opfattes som strings. En god måde at arbejde med det på er at gøre det så overskueligt for sig selv som muligt ved at opbygge dem så man altid kan se indholdet og ikke bare een lang string. Således kan en generel SQL-sætning opbygges:

```
dim sql
sql =    "SELECT attributnavn(e) " & _
    "FROM tabelnavn(e) " & _
    "WHERE betingelser;"
```

AD 1:
Her skrives alle navnene på de attributter du ønsker din forespørgsel skal vise på skærmen. Husk at separere alle navnene med kommaer. I denne afdeling kan du også lave beregningsudtryk. Til disse formål kan du evt. anvende udtrykkene SUM, MIN, MAX, COUNT, AVG, FIRST og LAST. Ønsker du at "omdøbe" en attributs navn til et andet er syntaksen således: "SELECT Name AS Navn". Attributter, der i tabellen er "Name", er i SQL-forespørgslen blevet til "Navn".

AD 2:
Her skriver du navne på alle de tabeller, der på en eller anden måde bruges i SQL-forespørgslen. I dette

afsnit er det også muligt at definere relationer mellem tabeller. Dette er dog forholdsvis svært at overskue hvis der er mange tabeller. Ved brug af flere tabeller adskilles de ved brug af komma ligesom i SELECT-afdelingen. Det er også i FROM-afdelingen at du kan arbejde med JOINs, dette vil jeg dog forklare i en senere artikel (simple joins vil blive behandlet i WHERE-afdelingen her!)

AD 3:
I denne afdeling skal du opstille de betingelser der måtte gælde for din SQL-forespørgsel. En betingelse kan fx være, at en bestemt værdi skal opfylde et bestemt krav. Fx kan du ønske at se en liste over alle der bor i postnummer 8000. Syntaksen kan fx være: WHERE Postnummer = 8000. I stedet for at bruge "=", kan du også bruge tegnene <, >, <=, >=, <>.

Ligeledes kan du vælge data, der befinder sig i et bestemt interval (BETWEEN). Dette kan bruges istedet for at sige WHERE Postnummer >= 8000 AND Postnummer <= 9000;

```
dim sql
sql =    "SELECT attributnavn(e) " & _
    "FROM tabelnavn1 " & _
    "WHERE Postnummer BETWEEN 8000 AND 9000;"
```

Dette vil så give dig et udtræk af postnummerene i intervallet 8000-9000.

I denne afdeling kan du også benytte dig af det reserverede ord NULL. En NULL-værdi er det samme som et tomt felt. Forestil dig, at du har lavet en adressedatabase over dine venner. I denne database er der en tabel, der har en attribut, der hedder Mobiltelefon. Da det ikke er alle dine venner der er den lykkelige ejer af et sådanne apparat, vil nogle af Mobiltelefon-felterne i din database være tomme. Professionelt hedder det, at feltet indeholder en NULL-værdi. Skal du fx søge på dine venner der ikke har mobiltelefon skal du så skrive "WHERE Mobiltelefon IS NULL". Hvad tror du der sker, hvis du skriver "WHERE Mobiltelefon IS NOT NULL"?

```
dim sql
sql =    "SELECT attributnavn(e) " & _
    "FROM tabelnavn1 " & _
    "WHERE Mobiltelefon IS NOT NULL;"
```

Bemærk hvis der er flere betingelser i din SQL-sætning, skal disse adskilles af enten et AND eller et OR.

```
dim sql
sql =    "SELECT attributnavn(e) " & _
    "FROM tabelnavn1 " & _
    "WHERE Mobiltelefon IS NOT NULL OR Mobiltelefon = 86000000 AND attributnavn
> 2;"
```

I ovenstående eksempel vil "Mobiltelefon = 86000000 AND attribut > 2" blive bindende, hvorimod at ved bruge af paranteser bliver resultatet anderledes, da indeholdet i parantesen bliver evalueret mod attributnavn > 2:

```
"WHERE (Mobiltelefon IS NOT NULL OR Mobiltelefon = 86000000) AND attributnavn >
2;"
```

I WHERE-afdelingen er det forholdsvis let at oprette relationer (simple joins) mellem de valgte tabeller (Dem valgte du i FROM-afdelingen). Der er dog begrænsninger i relationsmulighederne! En simpel relation oprettes på følgende måde: "WHERE tabelnavn1.primærnøglenavn = tabelnavn2.fremmednøglenavn". På den måde bliver primær- og fremmednøglerne fra de to forskellige tabeller knyttet sammen. Dette kan se således ud:

```
dim sql
sql =    "SELECT attributnavn(e) " & _
    "FROM tabelnavn1, tabelnavn2 " & _
    "WHERE Mobiltelefon IS NOT NULL AND " & _
    "tabelnavn1.primærnøglenavn = tabelnavn2.fremmednøglenavn;"
```

Bemærk at du ikke nødvendigvis behøver at vælge dine "relationsattributter" i din SELECT-afdeling - dette gælder også for resten af betingelserne i din WHERE-afdeling som man kan undlade i SELECT-afdelingen, hvis man ikke skal bruge dem til noget på hjemmesiden.

Sortering
Ønsker du at sortere de valgte data efter en bestemt orden, skal du benytte dig af udtrykket ORDER BY. Du kan sortere stigende (ASC) og faldende (DESC). Ønsker du at sortere efter flere felter, skal du blot skrive dem alle sammen efter hinanden (adskil med komma). Det attributnavn du skriver først er den attribut der sorteres primært. Det andet attributnavn sorteres sekundært osv.

```
dim sql
sql =    "SELECT attributnavn(e) " & _
    "FROM tabelnavn1, tabelnavn2 " & _
    "WHERE tabelnavn1.primærnøglenavn = tabelnavn2.fremmednøglenavn " & _
    "ORDER BY tabelnavn1.primærnøglenavn DESC, attributnavn ASC;"
```

Beregninger og GROUP BY
Bruger du beregningsudtryk i SELECT-afdelingen, skal du som oftest bruge GROUP BY til slut i SQL-sætningen.

```
dim sql
sql =    "SELECT attributnavn(e) " & _
    "FROM tabelnavn1 " & _
    "GROUP BY attributnavn(e) " & _
    "ORDER BY attributnavn DESC;"
```

GROUP BY er fx anvendelig ved en opsummering af forskellige forretningers priser (tænkt eksempel). Følgende eksempel kan måske være til hjælp:

Tabel: Forretning

Navn	Pris
Netto	500
Føtex	200
Føtex	250
Favorit	100
Netto	100
Brugsen	600
Føtex	400

```
|   Brugsen |       500 |
+-----------+-----------+
```

I SELECT-afdelingen vælger du at lave en SUM af priserne for nogle varer i et supermarked (SELECT SUM(pris) AS Pris). Da du kun vælger denne enkelte attribut, behøver du ikke at bruge GROUP BY. Resultatet af din forespørgsel vil blive det totale salg for **alle** forretninger.

```
dim sql
sql =    "SELECT SUM(pris) AS Pris " & _
    "FROM Forretning;"

+-----------+
|   Pris    |
+-----------+
|     2650  |
+-----------+
```

Skriver du derimod (SELECT Forretning, SUM(pris) AS Pris) vil du få en fejl, hvis du ikke har GROUP BY med. Fejlen opstår, da du på samme tid prøver at vise forretningernes navne og den samlede sum af priserne. Altså i venstre kolonne vil du gerne have et X antal forretningers navne skrevet ud, og i højre kolonne vil du gerne have én enkelt værdi; Den samlede pris for den enkelte butik. Nå man arbejder med tabeller skal der være lige mange rækker i de forskellige kolonner, derfor får du fejl.

Skriver du "GROUP BY Forretning", vil priserne blive summeret for hver enkelt forretning, og der er det samme antal rækker i hver kolonne - som der jo skal være.

```
dim sql
sql =    "SELECT Forretning, SUM(pris) AS Pris " & _
    "FROM Forretning " & _
    "GROUP BY Forretning " & _
    "ORDER BY Forretning DESC;"

+-----------+-----------+
|   Navn    | Pris      |
+-----------------------+
|   Netto   |       600 |
|   Føtex   |       850 |
|   Favorit |       100 |
|   Brugsen |      1100 |
+-----------+-----------+
```

Husk at alle *ikke*-beregninger du vælger i en GROUP BY skal være i din SELECT-afdeling også, som i eksemplet ovenover.

Husk på at du også altid kan udskrive din SQL-sætning på din ASP-side, så du kan teste den i dit lokale databasemiljø vha. cut'n'paste til MySQL, Access etc.:

```
dim sql
sql =    "SELECT Forretning, SUM(pris) AS Pris " & _
    "FROM Forretning " & _
    "GROUP BY Forretning;"
```

```
response.write sql
```

Forbindelse til databasen
Det lidt vanskelige når man arbejder med ASP-sider kan ofte være forbindelsen til ens database, der enten kan være svær at finde frem til stien på eller finde frem til den rette driver, som kan håndtere databasen.

Et udemærket sted at gå på udkig efter de såkaldte connectionstrings er på:
http://www.connectionstrings.com/

Et eksempel på en connection til en Access-database kommer her med SQL-kald. Hvert SQL-kald ned i databasen skulle gerne returnere et såkaldt recordset, hvor posterne er placeret således at vi kan kigge på dem i ASP-delen. Hvis vi ikke får et recordset kan det være fordi der ikke er nogen poster, der matcher vores forespørgsel eller at der ikke er nogen poster i databasen.

```
<%
Set MyConn = Server.CreateObject("ADODB.Connection")
MyConn.Open "Provider=Microsoft.Jet.OLEDB.4.0; Data
Source=f:\Inetpub\db\mindatabase.mdb"

dim sql
sql =    "SELECT Forretning, SUM(pris) AS Pris " & _
    "FROM Forretning " & _
    "GROUP BY Forretning;"
Set rsForretning = MyConn.Execute(SQL)

If rsForretning.EOF = True then 'finder ud af om recordsettet er tomt/fuld EOF
(End Of File)
    'der er IKKE noget i recordsettet
Else
    'der er noget i recordsettet
        While rsForretning.EOF = False 'loop gennem recordsettet mens EOF(End Of
File) er falsk
                Response.Write rsForretning("Forretning") & " " &
rsForretning("Pris") 'udskriv 1 post i recordset
                rsForretning.MoveNext 'flyt til næste post
        Wend 'løkken slutter
End If

'En god idé er altid at lukke ens adgang til databasen og sit recordset
rsForretning.Close 'luk recordset
Set MyConn = Nothing 'luk forbindelse til database
%>
```

Håber denne artikel har været med til at komme i gang med SQL og databaser i ASP.

Kommentar af gertnissen d. 09. Jan 2005 | 1

Giver en simpel begynder intro til SQL - select, where, simple funktioner og connect, uden at komme ind på f.eks. join, union, create, update mm.
Præcis hvad der loves.

Kommentar af cf560 d. 17. Jun 2008 | 2

Meget tyndarmet.

Kommentar af fnoop d. 25. Jun 2005 | 3

Kommentar af morteeart d. 03. Nov 2004 | 4

god guide, både til asp/sql, og til sql genneralt.

Kommentar af gruppe5 d. 14. Nov 2004 | 5

Kommentar af seb-boy d. 04. Aug 2005 | 6

Hmm, forstod nada,..