Denne guide er oprindeligt
udgivet på Eksperten.dk

COMPUTERWORLD

Introduktion til SQL queries

Denne artikel beskriver nogle forskellige muligheder i SQL queries. Eksemplerne skulle gerne være standard SQL og virke i alle databaser.

Den forudsætter kun minimalt kendskab til databaser og SQL og er klart for begyndere.

Skrevet den **16. Feb 2010** af **arne_v** I kategorien **Databaser / Generelt** |

Historie:
V1.0 - 18/08/2004 - original
V1.1 - 16/02/2010 - smårettelser

Introduktion

Q'et i SQL står for query (forespørgsel) og queries er da også en meget stor del af SQL sproget.

INSERT, UPDATE og DELETE statements er næsten altid ret simple, men SELECT statements kan blive meget komplekse.

Denne artikel vil gennemgå nogle af de gængse teknikker i queries.

Syntax

En SELECT statement består af følgende dele:

SELECT select liste
FROM tabel liste
[WHERE condition]
[GROUP BY group by liste]
[HAVING condition]
[ORDER BY order by liste]

De med [] markerede dele er optional.

Eksempel data

Alle eksemplerne vil bruge føgende simple data sæt:

medarbejder tabellen

```
id  navn                stilling        maanedsloen chef
1   Niels Nielsen       IT medarbejder  25000       3
2   Hans Hansen         IT medarbejder  30000       3
3   Jens Jensen         IT chef         37500
4   Anders Andersen     IT medarbejder  27500       3
5   Rasmus Rasmussen    IT trainee      22500       3
```

stilling tabellen

```
titel           overenskomst
IT chef         AC
IT medarbejder  HK
```

SQL sætninger til at create databasen med:

```
CREATE TABLE medarbejder (
  id INTEGER,
  navn VARCHAR(50),
  stilling VARCHAR(50),
  maanedsloen INTEGER,
  chef INTEGER,
  PRIMARY KEY (id)
);

INSERT INTO medarbejder VALUES (1,'Niels Nielsen','IT medarbejder',25000,3);
INSERT INTO medarbejder VALUES (2,'Hans Hansen','IT medarbejder',30000,3);
INSERT INTO medarbejder VALUES (3,'Jens Jensen','IT chef',37500,NULL);
INSERT INTO medarbejder VALUES (4,'Anders Andersen','IT medarbejder',27500,3);
INSERT INTO medarbejder VALUES (5,'Rasmus Rasmussen','IT trainee',22500,3);

CREATE TABLE stilling (
  titel VARCHAR(50),
  overenskomst VARCHAR(50),
  PRIMARY KEY (titel)
);

INSERT INTO stilling VALUES ('IT medarbejder','HK');
INSERT INTO stilling VALUES ('IT chef','AC');
```

Vælge kolonner

Du vælger hvilke kolonner der skal vises ved at angive dem i select listen. En * betyder alle felter.

```
SELECT *
FROM stilling;
```

```
titel           overenskomst
IT chef         AC
IT medarbejder  HK
```

```
SELECT overenskomst
FROM stilling;
```

```
overenskomst
AC
HK
```

```
SELECT navn,stilling
FROM medarbejder;
```

```
navn                stilling
Niels Nielsen       IT medarbejder
Hans Hansen         IT medarbejder
Jens Jensen         IT chef
Anders Andersen     IT medarbejder
Rasmus Rasmussen    IT trainee
```

Vælge rækker

Du vælger hvilke rækker der skal vises ved at angive en WHERE betingelse.

Du kan bruge normale udtryk (=, <>, <, >, <=, >=, AND, OR, NOT) og en speciel operator LIKE som sammenligner med wildcards (SQL bruger % som wildcard).

```
SELECT id, navn
FROM medarbejder
WHERE navn='Anders Andersen';
```

```
id  navn
4   Anders Andersen
```

```
SELECT id, navn
FROM medarbejder
WHERE maanedsloen > 25000;
```

```
id  navn
2   Hans Hansen
3   Jens Jensen
4   Anders Andersen´
```

```
SELECT id, navn
FROM medarbejder
WHERE maanedsloen > 25000 AND stilling <> 'IT chef';
```

```
id  navn
2   Hans Hansen
4   Anders Andersen
```

```
SELECT id, navn
FROM medarbejder
WHERE navn LIKE '%sen';
```

```
id  navn
1   Niels Nielsen
2   Hans Hansen
3   Jens Jensen
4   Anders Andersen
5   Rasmus Rasmussen
```

Vigtigt: feltnavn=NULL vil altid returnere false også selvom værdien er NULL. Man skal bruge feltnavn IS NULL.

Sortere output

Du sorterer de viste rækker med en ORDER BY på det felt der skal sorteres efter. Man kan tilføje ASC eller DESC for an angive stigende eller faldende sortering (default er stigende).

```
SELECT navn,maanedsloen
FROM medarbejder
ORDER BY maanedsloen;
```

```
navn                maanedsloen
Rasmus Rasmussen    22500
```

```
Niels Nielsen       25000
Anders Andersen     27500
Hans Hansen         30000
Jens Jensen         37500
```

```
SELECT navn,maanedsloen
FROM medarbejder
ORDER BY maanedsloen DESC;
```

```
navn                maanedsloen
Jens Jensen         37500
Hans Hansen         30000
Anders Andersen     27500
Niels Nielsen       25000
Rasmus Rasmussen    22500
```

Data fra flere tabeller

Du kan hente data fra flere tabeller. Det hedder i SQL sproget at joine tabeller.

Du kan lave forskellige slags joins. En inner join hvor der skal være en række i begge tabeller. En left outer join hvor der skal være en række i tabellen til venstre. En right outer join hvor der skal være en række i tabellen til højre.

Inner join med WHERE (den gamle måde):

```
SELECT medarbejder.navn,stilling.overenskomst
FROM medarbejder,stilling
WHERE medarbejder.stilling=stilling.titel;
```

```
navn                overenskomst
Niels Nielsen       HK
Hans Hansen         HK
Jens Jensen         AC
Anders Andersen     HK
```

Inner join med JOIN (den nye måde):

```
SELECT medarbejder.navn,stilling.overenskomst
FROM medarbejder INNER JOIN stilling ON medarbejder.stilling=stilling.titel;
```

```
navn                overenskomst
Niels Nielsen       HK
Hans Hansen         HK
Jens Jensen         AC
Anders Andersen     HK
```

Left join:

```
SELECT medarbejder.navn,stilling.overenskomst
FROM medarbejder LEFT JOIN stilling ON medarbejder.stilling=stilling.titel;
```

```
navn                overenskomst
Niels Nielsen       HK
Hans Hansen         HK
Jens Jensen         AC
Anders Andersen     HK
Rasmus Rasmussen
```

(bemærk at med mange tabeller så bruger man tabelnavn.feltnavn til at angive felter med)

Appende en query til en anden query

Man kan stable query resultater oven på hinanden med UNION.

```
(SELECT navn,stilling FROM medarbejder)
UNION
(SELECT navn,'fastansat' FROM medarbejder)
```

```
navn                stilling
Niels Nielsen       IT medarbejder
Hans Hansen         IT medarbejder
Jens Jensen         IT chef
Anders Andersen     IT medarbejder
Rasmus Rasmussen    IT trainee
Niels Nielsen       fastansat
Hans Hansen         fastansat
Jens Jensen         fastansat
Anders Andersen     fastansat
Rasmus Rasmussen    fastansat
```

(bemærk at man godt kan angive et beregnet udtryk i select listen)

Aggregerende funktioner og gruppering

Standard SQL har indbygget funktioner til at udregne minimum (MIN), maximum (MAX), antal (COUNT) og gennemsnit (AVG) for grupper.

```
SELECT COUNT(*) AS antal
FROM medarbejder;
```

```
antal
5
```

```
SELECT stilling,COUNT(*) AS antal,AVG(maanedsloen) AS gennemsnitsloen
FROM medarbejder
GROUP BY stilling;
```

```
stilling        antal   gennemsnitsloen
IT chef         1       37500
IT medarbejder  3       27500
IT trainee      1       22500
```

Bemærk at alle felter i select listen som ikke er aggregerende funktioner skal optræde i GROUP BY listen.

(bemærk endvidere at man kan give beregnede udtryk i select listen et felt navn)

Self join

Du kan godt riskikere at få brug for at joine en tabel med sig selv. Det kalder man for en self join.

```
SELECT m1.navn,m2.navn AS chef
FROM medarbejder AS m1,medarbejder AS m2
WHERE m1.chef=m2.id;
```

```
navn                chef
Niels Nielsen       Jens Jensen
Hans Hansen         Jens Jensen
Anders Andersen     Jens Jensen
Rasmus Rasmussen    Jens Jensen
```

(bemærk at ligesom med felter i select listen kan man også give tabeller andre navnw)

Subqueries

Man har mulighed for at neste queries.

Nestet løsning som virker i de fleste databaser:

```
SELECT navn
FROM medarbejder
WHERE stilling IN (SELECT titel FROM stilling WHERE overenskomst='HK');
```

```
navn
Niels Nielsen
Hans Hansen
Anders Andersen
```

Workaround med join som kan bruges i bl.a. MySQL 3.x og 4.0.x:

```
SELECT medarbejder.navn
FROM medarbejder,stilling
WHERE medarbejder.stilling=stilling.titel AND stilling.overenskomst='HK';
```

```
navn
Niels Nielsen
Hans Hansen
Anders Andersen
```

Videre

Det stærke ved SQL er at man kan kombinere alle elementerne i en SELECT statement.

Så i virkelighedens verden vil du ofte skulle kombinere mange af de teknikker
som er gennemgået her.

God SELECT lyst.

Kommentar af a1 d. 18. Aug 2004 | 1

hvis man er helt blank i SQL så er den en god start. du forklarer godt hvad Q'et står for, men ikke S & L, sql = structured query language. Der kunne også godt være lidt mere "kød" på, f.eks. kan en join i de fleste databaser laves alá: "SELECT m.navn, s.stilling FROM mederbejder m INNER JOIN stilling s ON m.fk_stillingID=s.ID" (jeg præfikser altid mine fremmednøgler (foreignkeys) med fk_, man kan jo så også diskutere om det ikke er bedre (hurtigere) at joine på et tal (INTEGER) eller en tekststregn, specielt når det enda er en VARCHAR). ;o)

Kommentar af simonvalter d. 27. Aug 2004 | 2

Gode eksempler

Kommentar af jakobgt d. 02. Mar 2005 | 3

Ganske udmærket. Jeg søgte en simpel forklaring på joins, og det må jeg sige den giver.

Kommentar af bak d. 20. Aug 2004 | 4

Fin artikel, lav du bare en opfølger :-)

Kommentar af wraber d. 05. Oct 2004 | 5

Kommentar af skizo_someone d. 27. Dec 2004 | 6

utrolig lærerig og velskrevet artikel

Kommentar af steenolsen1 d. 31. Jan 2005 | 7

Rigtig god oversigt! Den er printet ud og ligger som "opslagsværk" :-)

Kommentar af brianmilan d. 11. Oct 2005 | 8

Den giver svar på mange ting, har printet den ud til hjælp :-)

Kommentar af weaponx d. 21. Aug 2004 | 9

Fin artikel. Det ville være rart hvis den også medtog INSERT og ALTER

Kommentar af wicez (nedlagt brugerprofil) d. 07. May 2005 | 10

Rigtig god artikel, den gennemgår alle de basale og lettere avancerede "funktioner" i MySQL

Kommentar af abeass d. 18. Oct 2004 | 11

Kommentar af ohmygod d. 17. Dec 2004 | 12

Flot =)

Kommentar af showsource d. 15. Feb 2008 | 13