SQL-forespørgsler

En SQL-forespørgsel (på engelsk en query) er en kommando der begynder med SELECT og som returnerer data som er forespørgslens resultat.

En forespørgsel vil returnere et resultat, der består af ingen, en eller flere rækker. Disse rækker indeholder en eller flere kolonner.

En forespørgsel består af følgende

```
SELECT <kolonner>
FROM <tabel>
WHERE <betingelse for rækker>
```

SELECT angiver hvilke kolonner der skal medtages i hver af de rækker som returneres.
FROM angiver hvilken/hvilke tabeller der skal forespørges i.
WHERE angiver betingelse/betingelser for hvilke rækker der skal medtages i resultatet. WHERE-delen kan udelades, og dette vil resultere i at alle rækker i tabellen bliver medtaget.

Den korteste SQL-forespørgsel der kan angives

```
SELECT *
FROM Kunde
```

Dette giver resultatet som vist ovenfor i Data i tabellen Kunde, som er hele tabellens indhold.

SELECT

SELECT *
betyder medtag alle kolonner i de resulterende rækker.

SELECT <kolonnenavn>, <kolonnenavn>, . . .
betyder medtag de angivne kolonner i de resulterende rækker.

SELECT <tabelnavn>.*
betyder at alle kolonner i den kvalificerende table skal med. Dette bruges når der hentes fra flere tabeller. Udtrykket kan indgå i den kommaseparede liste af kolonnenavne.

SELECT <tabelnavn>.<kolonnenavn>
Betyder at det er den angivne kolonne i den angivne tabel, der skal med. Bruges når der er navnesammenfald imellem kolonner i to eller flere tabeller, som der forespørges i.

SELECT <kolonnenavn> AS <overskrift/nyt kolonnenavn>
Betyder at der gives en ny overskrift til kolonnen. Det er egentlig et alias sådan at kolonnen får et andet navn i resultatet. Hvis resultatet bruges i et program skal dette nye navn på kolonnen benyttes for at få fat i kolonnen.

SQL-forespørgsler

Der må gerne være ”ulovlige” tegn i det nye navn, såsom mellemrum, hvis navnet omgives af ” ” eller [ ]. Dette gælder i øvrigt al navngivning når der oprettes nye databaseobjekter.

Eksempler.
Eksemplerne er for at belyse kolonner. Der skal ses bort fra at resultatet i øvrigt ikke giver fuld mening, når der hentes fra flere tabeller uden at der er en WHERE del i kommandoen.

```
SELECT ForNavn, EfterNavn
FROM Kunde
```

Returnerer alle rækker i tabellen Kunde og hver med de angivne to kolonner.

ForNavn	EfterNavn
Jan	Johansen
Kasper	Kaspersen
Niels	Hansen
Amir	Jensen
Ingvar	Nilsson
. . .	

```
SELECT ForNavn, EfterNavn, Ordre.*
FROM Kunde, Ordre
```

Returnerer rækker med to kolonner fra tabellen Kunde og alle kolonner fra tabellen Ordre

ForNavn	EfterNavn	OrdreNr	Dato	Rabat	Total	KundeNr
Jan	Johansen	10000	2001-10-20	5.00	126,50	1003
Kasper	Kaspersen	10000	2001-10-20	5.00	126,50	1003
Niels	Hansen	10000	2001-10-20	5.00	126,50	1003
Amir	Jensen	10000	2001-10-20	5.00	126,50	1003
Ingvar	Nilsson	10000	2001-10-20	5.00	126,50	1003
. . .						

Når der angives et kolonnenavn, som findes i flere af de berørte tabeller, vil der komme en fejlmeddelelse om at kolonne-angivelsen er tvetydig. I nedenstående kommando er kolonnen KundeNr tvetydig, da den findes i begge tabeller. Dette løses med at kvalificere kolonnenavnet med den tabel den ønskes fra. Altså Kunde.KundeNr

SQL-forespørgsler

```
SELECT Kunde.KundeNr, ForNavn, EfterNavn, Ordre.*
FROM Kunde, Ordre
```

KundeNr	ForNavn	EfterNavn	OrdreNr	Dato	Rabat	Total	KundeNr
1000	Jan	Johansen	10000	2001-10-20	5.00	126,50	1003
1001	Kasper	Kaspersen	10000	2001-10-20	5.00	126,50	1003
1002	Niels	Hansen	10000	2001-10-20	5.00	126,50	1003
1003	Amir	Jensen	10000	2001-10-20	5.00	126,50	1003
1004	Ingvar	Nilsson	10000	2001-10-20	5.00	126,50	1003
. . . .							

I følgende forespørgsel er der medtaget kolonnen KundeNr fra begge tabeller. Da overskriften for begge kolonner, nemlig kolonnenavnet, i resulatet vil være det samme, kunne det være ønskeligt at give en ny overskrift. Dette gøres med aliaset ”AS”.

```
SELECT Kunde.KundeNr AS "Kunde.KundeNr", ForNavn, EfterNavn,
    OrdreNr, Ordre.KundeNr AS "Ordre.KundeNr"
FROM Kunde, Ordre
```

Kunde.KundeNr	ForNavn	EfterNavn	OrdreNr	Ordre.KundeNr
1000	Jan	Johansen	10000	1003
1001	Kasper	Kaspersen	10000	1003
1002	Niels	Hansen	10000	1003
1003	Amir	Jensen	10000	1003
1004	Ingvar	Nilsson	10000	1003

FROM

FROM efterfølges af den/de tabeller, der skal hentes data fra.
Når der angives flere tabeller, vil der som udgangspunkt dannes alle de kombinationer af rækker som er muligt fra tabellerne. Hvis der angives to tabeller, den ene med 12 rækker og den anden med 6 rækker, vil der dannes 12*6 = 72 rækker som resultat. (Det er så op til WHERE-delen af udvælge de ønskede rækker.)

FROM <tabelnavn>, <tabelnavn>, . . .

Se ovenstående eksempler

Det er muligt i FROM-delen at give tabellerne alias-navne, som der så kan henvises til i den øvrige del af kommanoen.
FROM <tablenavn> AS <aliasnavn>

SQL-forespørgsler

Eksempel:

```
SELECT Kunde.KundeNr, ForNavn, EfterNavn, O.*
FROM Kunde, Ordre AS O
```

KundeNr	ForNavn	EfterNavn	OrdreNr	Dato	Rabat	Total	KundeNr
1000	Jan	Johansen	10000	2001-10-20	5.00	126,50	1003
1001	Kasper	Kaspersen	10000	2001-10-20	5.00	126,50	1003
1002	Niels	Hansen	10000	2001-10-20	5.00	126,50	1003
1003	Amir	Jensen	10000	2001-10-20	5.00	126,50	1003
1004	Ingvar	Nilsson	10000	2001-10-20	5.00	126,50	1003
. . .							

Ovenstående returnerer rækker med to kolonner fra tabellen Kunde og alle kolonner fra tabellen Ordre.

Det interessante i resultatet er kolonnerne og ikke rækkerne.
Se bort fra indholdet, som ikke giver nogen mening, da alle kunder udskrives sammen alle ordrer.
Her er kun medtaget de første rækker, hvor det er den første Ordre der er kombineret med alle Kunder.

KundeNr for Kunder og for Ordrer passer overhovedet ikke sammen, da alle kunder udskrives sammen med alle ordrer. Det leder til WHERE-delen hvor dette problem løses.

WHERE

WHERE-delen angiver en filtrering af rækker. Dette gøres ved at en eller flere betingelser afgør hvilke rækker der skal medtages i resultatet.

Eksempel:

```
SELECT * FROM Kunde
WHERE EfterNavn = 'Kaspersen'
```

KundeNr	ForNavn	EfterNavn	Tlf	Mail	Anciennitet
1001	Kasper	Kaspersen	34532455	kas@kasper.dk	2
1011	Jytte	Kaspersen	34532455	jyt@kaspersen.dk	3

Problemet fra tidligere med det såkaldte cartetiske product, hvor alle rækker i hver tabel kombineres med alle rækker i hver tabel, når der hentes fra flere tabeller, skal løses.
Det er kun relevant at se hver Kunde sammen med de Ordrer som har et KundeNr svarende til Kundes KundeNr.

SQL-forespørgsler

Det gøres ved i WHERE-delen at angive Kunde.KundeNr = Ordre.KundeNr.

```
SELECT Kunde.KundeNr, ForNavn, EfterNavn, Ordre.*
FROM Kunde, Ordre
WHERE Kunde.KundeNr = Ordre.KundeNr
```

KundeNr	ForNavn	EfterNavn	OrdreNr	Dato	Rabat	Total	KundeNr
1001	Kasper	Kaspersen	10003	2003-03-08	3.00	664,35	1001
1001	Kasper	Kaspersen	10009	2007-10-13	5.00	600,00	1001
1002	Niels	Hansen	10001	2002-01-21	2.00	1450,00	1002
1002	Niels	Hansen	10002	2003-12-09	2.00	340,00	1002
1003	Amir	Jensen	10000	2001-10-20	5.00	126,50	1003
1003	Amir	Jensen	10004	2003-04-23	5.00	780,00	1003
1003	Amir	Jensen	10008	2006-10-23	5.00	430,00	1003
1005	Hans Jørgen	NULL	10010	2008-02-12	10.00	170,00	1005
1006	Dan	Petersen	10011	2009-01-10	15.00	356,50	1006
1007	Ida	Pedersen	10005	2004-06-07	4.00	100,00	1007
1009	Abdul	Wahid	10006	2004-09-19	2.00	200,00	1009
1012	Jytte	Nybro	10007	2005-11-25	5.00	1230,75	1012

Operatorer der kan bruges i WHERE-filtreringen

Operatorerne bruges til at sammenligne en kolonnes indhold med en værdi eller et område af værdier

=, <>, !=, <, >, <=, >=, !<, !>,
BETWEEN . . . AND . . .
IS NULL, IS NOT NULL

Eksempler:

```
SELECT * FROM Kunde
WHERE KundeNr = 104
```

KundeNr	Fornavn	Efternavn	Telefon	E-mail	Anciennitet
104	Ingvar	Nilsson	39453245	ing@nilsson.dk	3

SQL-forespørgsler

```
SELECT * FROM Kunde
WHERE KundeNr > 109
```

KundeNr	Fornavn	Efternavn	Telefon	E-mail	Anciennitet
110	Helle	Knabstrup	31425332	hel@knap.dk	4
111	Jytte	Kaspersen	34532455	jyt@kaspersen.dk	3
112	Jytte	Nybro	34532455	jyt@nybroe.dk	3

```
SELECT * FROM Kunde
WHERE KundeNr BETWEEN 100 AND 102
```

KundeNr	Fornavn	Efternavn	Telefon	E-mail	Anciennitet
100	Jan	Johansen	12341234	jaj@tec.dk	1
101	Kasper	Kaspersen	34532455	kas@kasper.dk	2
102	Niels	Hansen	12324551	niels@hansen.dk	1

NULL

Det kan ikke lade sig gøre at sammenligne med NULL. Alle sammenligninger med NULL vil returnere falsk. Selv NULL = NULL vil være falsk. NULL betyder udefineret og kan derfor betyde alt og intet.
For at filtrere på NULL skal der bruges IS NULL eller IS NOT NULL.

```
SELECT * FROM Kunde
WHERE EfterNavn IS NULL
```

KundeNr	Fornavn	Efternavn	Telefon	E-mail	Anciennitet
105	Hans Jørgen	NULL	83453245	hj@knudsen.dk	4

Avanceret filtrering

AND og OR imellem flere betingelser
IN, NOT IN <mængde>
EXISTS, NOT EXISTS <resultat af forespørgsel>
ALL (xx < ALL <mængde af samme type som xx>)
ANY (xx < ANY <mængde af samme type som xx>)

SQL-forespørgsler

Eksempler:

```
SELECT * From Kunde
Where KundeNr > 104
AND Anciennitet = 1
```

KundeNr	Fornavn	Efternavn	Telefon	E-mail	Anciennitet
108	Ahmed	Suleman	41532455	ahm@sule.dk	1
109	Abdul	Wahid	13532455	abd@wahid.dk	1

```
SELECT * From Kunde
Where EfterNavn = 'Petersen'
OR EfterNavn = 'Pedersen'
```

KundeNr	Fornavn	Efternavn	Telefon	E-mail	Anciennitet
106	Dan	Petersen	31452455	dan@petersen.dk	5
107	Ida	Pedersen	45312455	ida@pedersen.dk	2

Eksempler på EXISTS og NOT EXISTS samt ANY og ALL har med underforespørgsler/subqueries at gøre og venter til dette afsnit.

SQL-forespørgsler

Filtrering med wildcard

Disse bruges ved sammenligning af tekster. Når der sammenlignes med en nøjagtig tekst bruges ”=”. Når der sammenlignes med dele af tekster sammen med wild-cards, skal der bruges LIKE.

LIKE
% - ingen, en eller flere vilkårlige karakterer.
_ - en vilkårlig karakter
’[JN]’ - karakteren J eller N
’[^JN] - ikke karakteren J eller N
’[0-9]’ - karakteren, som har en af værdierne fra 0 til 9

Eksempler:

Alle Efternavne der begynder med ”P”

```
SELECT * FROM Kunde
WHERE EfterNavn LIKE 'P%'
```

KundeNr	Fornavn	Efternavn	Telefon	E-mail	Anciennitet
106	Dan	Petersen	31452455	dan@petersen.dk	5
107	Ida	Pedersen	45312455	ida@pedersen.dk	2

Alle Efternavne hvor andet bogstav er “e”

```
SELECT * FROM Kunde
WHERE EfterNavn LIKE '_e%'
```

KundeNr	Fornavn	Efternavn	Telefon	E-mail	Anciennitet
103	Amir	Jensen	56345322	amir@jensen.dk	2
106	Dan	Petersen	31452455	dan@petersen.dk	5
107	Ida	Pedersen	45312455	ida@pedersen.dk	2

SQL-forespørgsler

Alle ”sen”-navne.

```
SELECT * FROM Kunde
WHERE EfterNavn LIKE '%sen'
```

KundeNr	Fornavn	Efternavn	Telefon	E-mail	Anciennitet
100	Jan	Johansen	12341234	jaj@tec.dk	1
101	Kasper	Kaspersen	34532455	kas@kasper.dk	2
102	Niels	Hansen	12324551	niels@hansen.dk	1
103	Amir	Jensen	56345322	amir@jensen.dk	2
106	Dan	Petersen	31452455	dan@petersen.dk	5
107	Ida	Pedersen	45312455	ida@pedersen.dk	2
111	Jytte	Kaspersen	34532455	jyt@kaspersen.dk	3

Alle hvor “J” eller “S” er det første bogstav.

```
SELECT * FROM Kunde
WHERE EfterNavn LIKE '[JS]%'
```

KundeNr	Fornavn	Efternavn	Telefon	E-mail	Anciennitet
100	Jan	Johansen	12341234	jaj@tec.dk	1
103	Amir	Jensen	56345322	amir@jensen.dk	2
108	Ahmed	Suleman	41532455	ahm@sule.dk	1

Alle hvor næstsidste bogstav ikke er “a”, “e” eller “i”.

```
SELECT * FROM Kunde
WHERE EfterNavn LIKE '%[^aei]_'
```

KundeNr	Fornavn	Efternavn	Telefon	E-mail	Anciennitet
104	Ingvar	Nilsson	39453245	ing@nilsson.dk	3
110	Helle	Knabstrup	31425332	hel@knap.dk	4
112	Jytte	Nybro	34532455	jyt@nybroe.dk	3

SQL-forespørgsler

DISTINCT

DISTINCT viser alle værdier i den aktuelle kolonne, men kun i et eksemplar hver.

Følgende kommando viser i KundeNr der er på Ordrene.

SELECT KundeNr FROM Ordre

KundeNr
103
102
102
101
103
107
109
112
103
101
105
106

Følgende kommando viser de forskellige KundeNr der er på Ordrene. Altså hvilke kunder der har afgivet Ordrer.

SELECT DISTINCT KundeNr FROM Ordre

KundeNr
101
102
103
105
106
107
109
112

SQL-forespørgsler

Sortering

Sortering foregår ved at afslutte SELECT-sætningen med ORDER BY <kolonne>. Kolonne kan være et eller flere kolonner angivet med navn eller nummer i SELECT-rækkefølgen.
Hvis der angives flere kolonner, sorteres der først efter den først angivne. Hvis der er ens værdier i denne kolonne, sorteres der derefter på næste kollen osv.
Der sorteres default i stigende rækkefølge, hvilket kan angives med ASCENDING. Faldende rækkefølge angives med DESCENDING.

ORDER BY <kolonnenavn>
ORDER BY <kolonnenavn>, <kolonnenavn>, ...
ORDER BY <kolonnenr>
ORDER BY <kolonnenr>, <kolonnenr>, ...
ORDER BY <kolonne> DESC/ASC – descending/ascending - default ascending

Eksempler.

Sorter på en kolonne, her Efternavn.

```
SELECT EfterNavn, ForNavn FROM Kunde
ORDER BY EfterNavn
```

Eller

```
SELECT ForNavn, EfterNavn FROM Kunde
ORDER BY 2
```

Fornavn	Efternavn
Hans Jørgen	NULL
Niels	Hansen
Amir	Jensen
Jan	Johansen
Kasper	Kaspersen
Jytte	Kaspersen
Helle	Knabstrup
Ingvar	Nilsson
Jytte	Nybro
Ida	Pedersen
Dan	Petersen
Ahmed	Suleman
Abdul	Wahid

SQL-forespørgsler

Sorter på to kolonner, her Efternavn og Fornavn.

```
SELECT ForNavn, EfterNavn FROM Kunde
ORDER BY EfterNavn, ForNavn
```

Eller

```
SELECT ForNavn, EfterNavn FROM Kunde
ORDER BY 2,1
```

Fornavn	Efternavn
Hans Jørgen	NULL
Niels	Hansen
Amir	Jensen
Jan	Johansen
Jytte	Kaspersen
Kasper	Kaspersen
Helle	Knabstrup
Ingvar	Nilsson
Jytte	Nybro
Ida	Pedersen
Dan	Petersen
Ahmed	Suleman
Abdul	Wahid

Sort faldende på Efternavn og stigende på Fornavn.

```
SELECT ForNavn, EfterNavn FROM Kunde
ORDER BY EfterNavn DESC, ForNavn ASC
```

Fornavn	Efternavn
Abdul	Wahid
Ahmed	Suleman
Dan	Petersen
Ida	Pedersen
Jytte	Nybro
Ingvar	Nilsson
Helle	Knabstrup
Jytte	Kaspersen
Kasper	Kaspersen
Jan	Johansen
Amir	Jensen
Niels	Hansen
Hans Jørgen	NULL

SQL-forespørgsler

Hent et antal af de første resultater

Ved hjælp af TOP <antal> kan der udvælges et antal af de første rækker i resultatet.

```
SELECT TOP 10 * FROM Vare
```

VareNr	Navn	Beskrivelse	Pris	Lager
1	Skummetmælk 1	Øko Thiese	7,80	200
2	Skummetmælk 2	Fritgående Thiese	6,80	200
3	Skummetmælk 3	Billig Arla	5,80	200
4	Mel Fin	Øko Bagerens	12,00	100
5	Mel Grov	Kværnens billege	7,50	120
6	Ris Natur	Øko Fra Bangladesh	17,80	20
7	Ris Poleret	Den knap så sunde	7,80	30
8	Bulgur	Knækket hvede	25,99	45
9	Sild marineret	Fra Aggersund	25,00	15
10	Sild carry	Fra Harboøre og Indien	25,00	17

Når der hentes et antal af de første rækker i resultatet, vælges rækkerne ud efter at det øvrige resultat er fundet og evt. sorteret.

```
SELECT TOP 10 * FROM Vare
ORDER BY Pris
```

VareNr	Navn	Beskrivelse	Pris	Lager
3	Skummetmælk 3	Billig Arla	5,80	200
13	Affaldspose	Med lukning	5,80	48
2	Skummetmælk 2	Fritgående Thiese	6,80	200
14	Frysepose 1	1 liter plast	7,00	30
15	Frysepose 2	2 liter plast	7,00	25
17	Kiks Tuc	Den der knaser	7,00	55
5	Mel Grov	Kværnens billege	7,50	120
7	Ris Poleret	Den knap så sunde	7,80	30
1	Skummetmælk 1	Øko Thiese	7,80	200
24	Rapsolie	Dansk	7,80	40

SQL-forespørgsler

SOUNDEX()

Med funktionen soundex() kan der filtreres på tekster der lyder lignende det angivne.
Tabellen Person indeholder følgende.

Navn
Martin
Martha
Nortam
Morten
Mortem
Martine
Martines
Martinus
Magnus
Wassim

I følgende hentes de navne der lyder som Martin.

```
SELECT * FROM Person WHERE soundex(Navn) LIKE soundex('Martin')
```

Navn
Martin
Morten
Mortem
Martine
Martines
Martinus

SQL-forespørgsler

I følgende hentes de navne der ikke lyder som Martin.

```
SELECT * FROM Person WHERE soundex(Navn)NOT LIKE soundex('Martin')
```

Navn
Martha
Nortam
Magnus
Wassim

Konstruering og beregning af kolonner i SELECT

I SELECT kan der konstrueres nye kolonner i resultatet. Dette kan ske ved beregninger og sammensætning af tekster, hvori der indgår eksisterende kolonner og konstanter. Det er muligt at skrive en SELECT kommando uden at der angives en FROM-del, hvis der kun bruges konstanter og funktionskald.

Eksempler.

Der bruges en tekst-konstant alene

```
SELECT 'Hello World' AS Hilsen
```

Hilsen
Hello World

Der laves en simpel beregning med to konstanter

```
SELECT 1000 * 1.25 AS "Pris med moms"
```

Pris med Moms
1250.00

En tekst sættes sammen med en datetime, som er castet til en tekst.

```
SELECT 'Nu er det ' + cast(getdate() AS varchar(20)) As 'Lige nu'
```

Lige nu
Nu er det Nov 6 2010 3:03PM

SQL-forespørgsler

Der udføres en beregning med to kolonner som input på tabellen Ordre.

```
SELECT OrdreNr, SubTotal, Rabat, SubTotal * Rabat AS Total
FROM Ordre
```

OrdreNr	SubTotal	Rabat	Total
10000	425,94	5.00	2129.700000
10001	200,00	2.00	400.000000
. . .			

Et pænere resultat ved at caste beregningen til decimal-typen hvor antal decimaler sættes til 2.

```
SELECT OrdreNr, SubTotal, Rabat,
  cast((SubTotal * Rabat) AS Decimal(7,2)) AS Total
FROM Ordre
```

OrdreNr	SubTotal	Rabat	Total
10000	425,94	5.00	2129.70
10001	200,00	2.00	400.00
. . .			

```
SELECT ForNavn + ' ' + EfterNavn AS Navn FROM Kunde
```

Navn
Jan Johansen
Kasper Kaspersen
Niels Hansen
Amir Jensen
Ingvar Nilsson
NULL
Dan Petersen
. . .

SQL-forespørgsler

Det viser sig at når NULL indgår i en konstruktion, bliver resultatet NULL. Her er efternavnet NULL på 'Hans Jørgen', hvilket gør at hele navnet bliver NULL. Dette gælder også ved beregninger og sammenligninger

```
SELECT ForNavn + ' ' + isNull(EfterNavn, '<Efternavn>') AS Navn FROM Kunde
```

Navn
Jan Johansen
Kasper Kaspersen
Niels Hansen
Amir Jensen
Ingvar Nilsson
Hans Jørgen <Efternavn>
Dan Petersen
. . .

Eksempel på Employee tabellen

.

```
SELECT FName + ' ' + MInit + ' ' + LName AS Navn,
  right('    ' + convert(varchar(10),cast((sal + isnull(comm, 0))
  AS decimal(10,2))),8) AS Løn
FROM Emp
```

Fornavn	Efternavn
Harald H Smith	800.00
Dave J Allen	1900.00
Samuel D Ward	1750.00
Jack K Jones	2975.00
Jeans J Martin	2650.00
Horace O Blake	2850.00
Terry A Clerk	2450.00
Taylor R Scott	3000.00
James L KING	5000.00
Tina E Turner	1500.00
John R Ford	3000.00
Marcus G Miller	1300.00
Harald H Smith	800.00

SQL-forespørgsler

Første kolonne i forspørgslen er en sammensætning af 3 tekst-felter.
Anden kolonne består af summen af felterne Sal og Comm, altså løn og kommission. Da Comm i nogle rækker er null, må de erstattes af værdien 0 for at kunne lægges sammen. Null erstattes med 0 ved hjælp af funktionen isnull().
Denne sum castes med metoden cast til en decimal-type med to decimaler.
Derefter konverteres decimalværdien til tekst og sættes sammen med 4 spaces i venstre side og der udtages de 8 karakterer fra højre for at højre-stille tallene pænt. (Dette i stedet for at indsætte padding-spaces, så alle fylder det samme, hvilket man åbenbart ikke kan)