MS SQL & Windows Server

RASMUS JUEL NIELSEN

Datatekniker med speciale i Programmering

Indholdsfortegnelse

MS SQL & Windows Server
Rasmus Juel Nielsen

Dagbog

Dag 1: Jeg starter med at oprette en ny virtuel maskine i Hyper-V ved brug af iso filerne til Windows server 2019. Finder ud af at iso-filerne ligger på et fælles netværksdrev efter et godt stykke tid og en del bakseri med at downloade dem fra nettet. SQL serveren og databasen bliver installeret, og starter stille og roligt med at undersøge hvordan SQL kodning fungerer.

Dag 2: Skriver koden færdig uden så mange problemer, Primary og Foreign keys oprettes lidt febrilsk ved at prøve forskellige metoder indtil det virker, men er stadig lidt usikker på hvordan det fungerer.

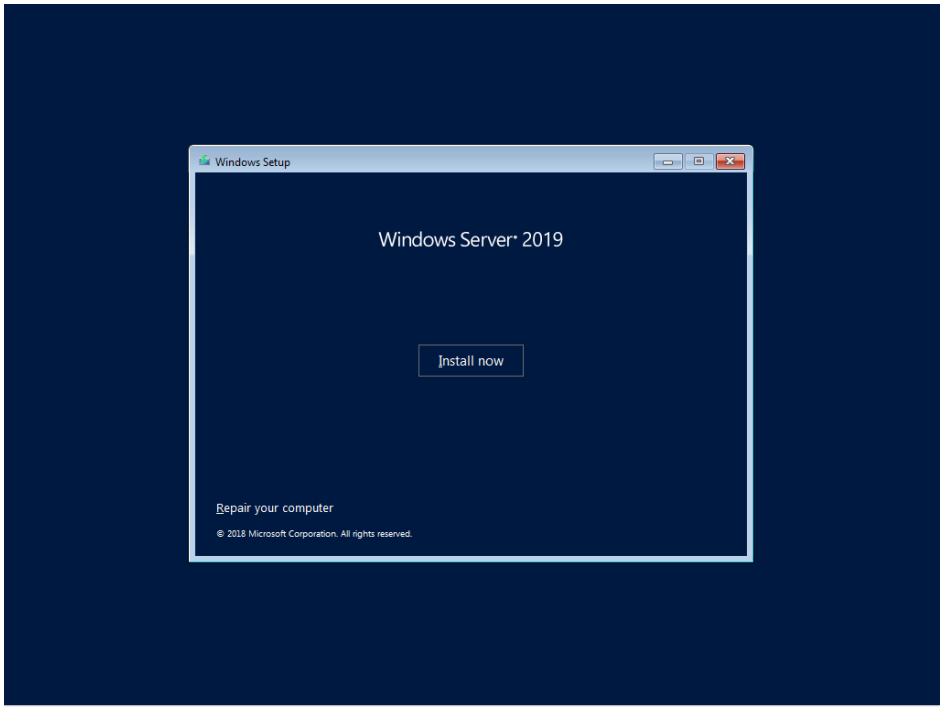
Dag 3: Roder lidt med select og join, og der er lige noget med at joine flere tabeller der driller lidt, men finder hurtigt svar på nettet. Begynder så småt på rapporten.

Dag 4: Færdiggør rapport.

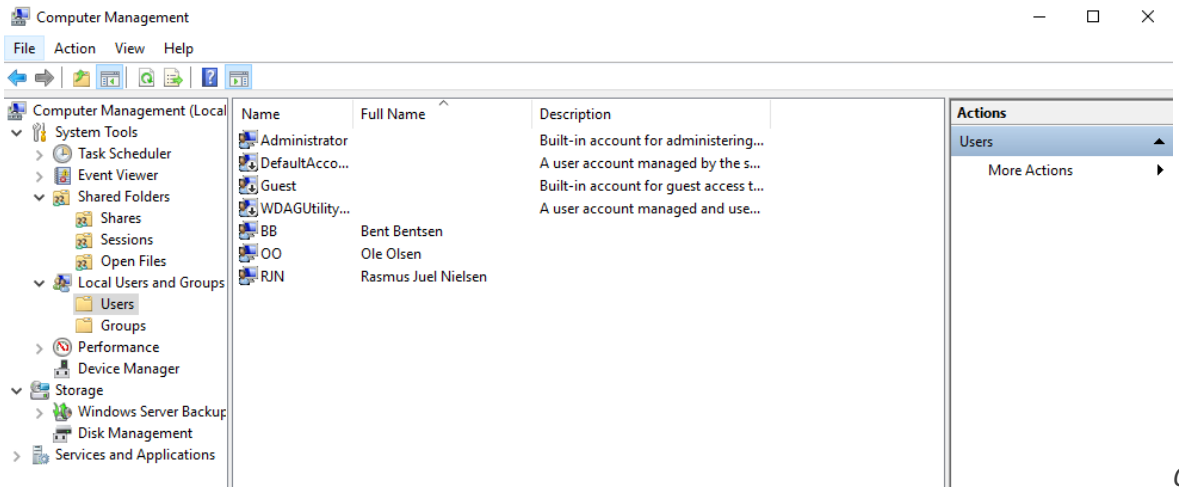
Indledning

Opgaven tager udgangspunkt i programmeringssproget SQL (Structured Query Language), som er et af de mest udbredte sprog til brug af relationelle databaser og til udførsel af en lang række funktioner på baggrund af databasernes data. Opgaven går ud på at oprette en række tabeller indeholdende data som kan forbindes til nye tabeller alt efter hvilken data der efterspørges, ved brug af select og join funktioner. Det hele foregår på en virtuel windows server 2019 i Hyper-V, hvorpå selve databasen, SQL Developer, og SSMS (SQL Server Management Studio) installeres.

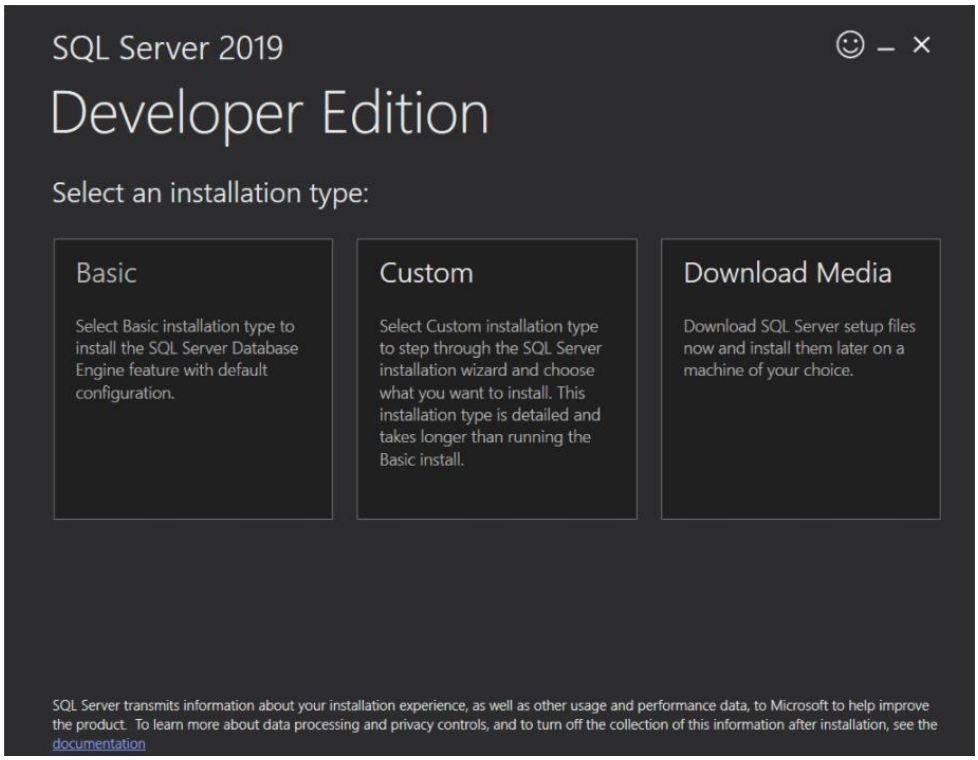
Installationsproces



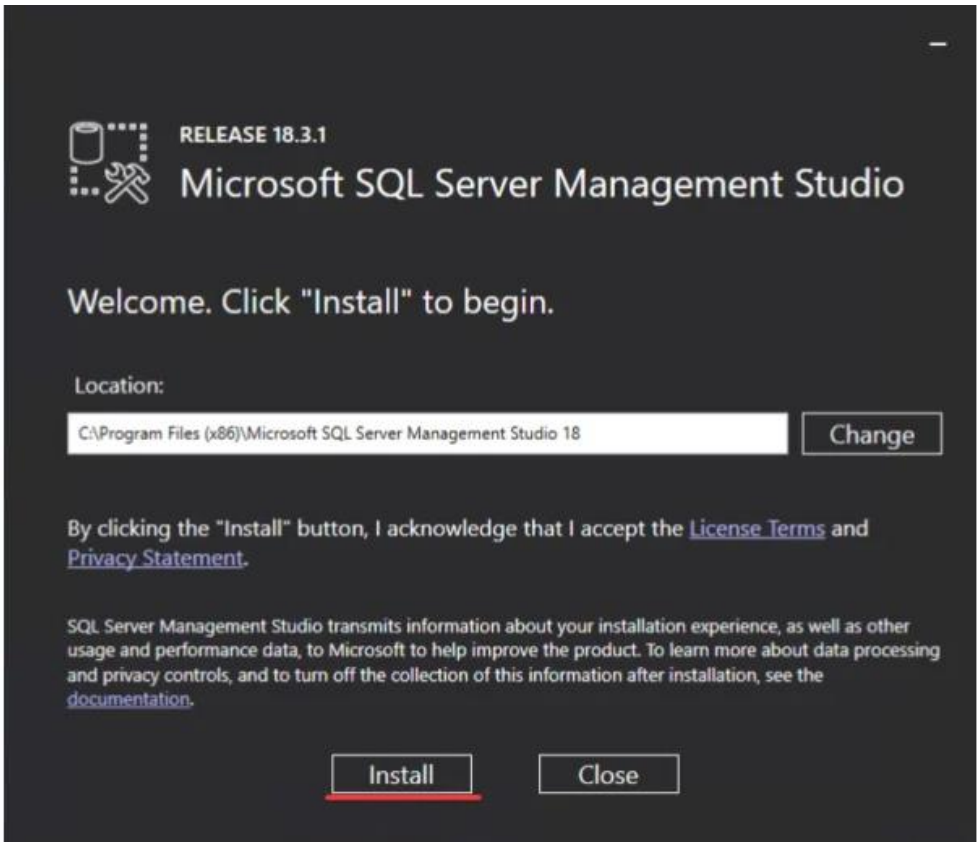
Windows server 2019 installeres i Hyper-V.



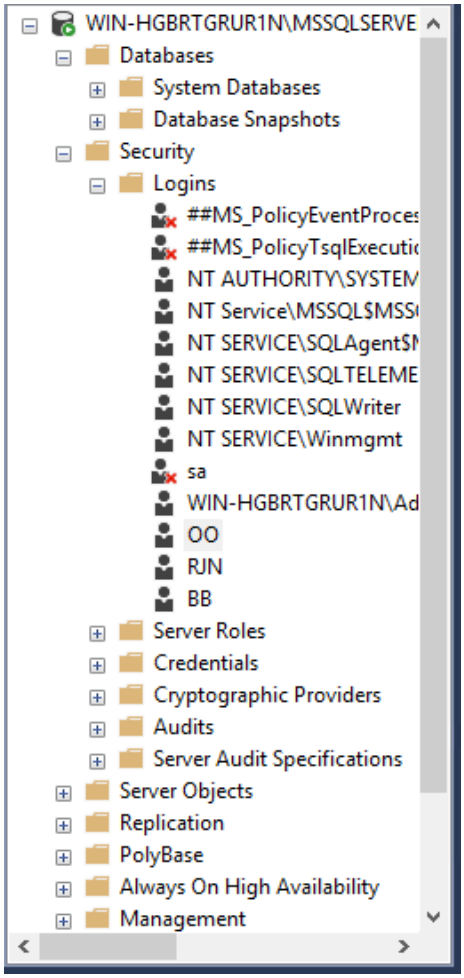
Opretter 3 nye brugere på Windows 2019 serveren (BB, OO og RJN).



SQL Developer installeres. Der vælges Basic installation.



SSMS (SQL Server Management Studio) installeres.

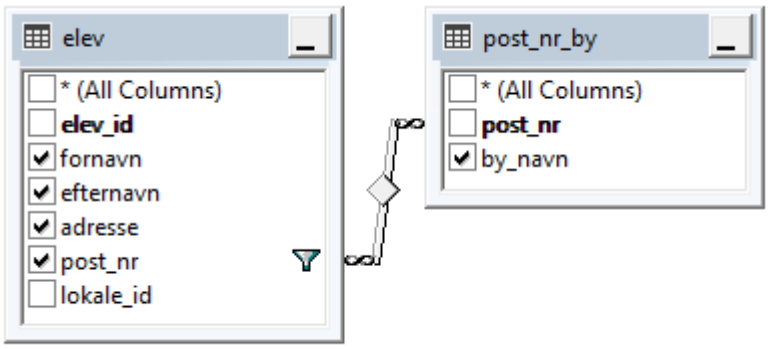


3 brugere oprettes på SQL serveren (OO, RJN og BB).

Select og join

Eksempel 1

```
select fornavn, efternavn, adresse, elev.post_nr, by_navn
from elev
join post_nr_by
on elev.post_nr=post_nr_by.post_nr
```



Variablerne fornavn, efternavn, adresse, post_nr og by_navn vælges, som det output der ønskes i den nye tabel. ”from elev” indikerer at det er data fra elev tabellen der ønskes at blive linket. By_navn variablen hentes fra post_nr_by tabellen, og kan lade sig gøre ved brug af join funktionen. De linkes på baggrund variablen post_nr, som viser når post_nr i elev tabellen er lig post_nr i post_nr_by tabellen, og kan dermed fortælle hvilken by der tilhører hver enkelt elevs postnummer, hvis det findes i databasen.

Tabellen kommer til at se således ud:

	fornavn	efternavn	adresse	post_nr	by_navn
1	Bo	Andersen	Gammel Byvej 12	2650	Hvidovre
2	Frederikke	Hansen	Amager Boulevard 5	2300	København S
3	Jens	Mikkelsen	Lily Brogbergs Vej 17	2500	Valby
4	Philip	Mortensen	Brunevang 90	2610	Rødovre
5	Kasper	Frederiksen	Bryggertorvet 32	3650	Ølstykke
6	Milla	Jørgensen	Virum Torv 25	2830	Virum
7	Fie	Knudsen	Allen 85	2770	Kastrup
8	Henrik	Madsen	Lily Brobergs Vej 53	2500	Valby
9	Rasmus	Nielsen	Nøddehaven 58	2500	Valby

Man kan yderligere tilføje et ”where” statement, så der kun vises f.eks. de elever som bor i postnummeret 2500:

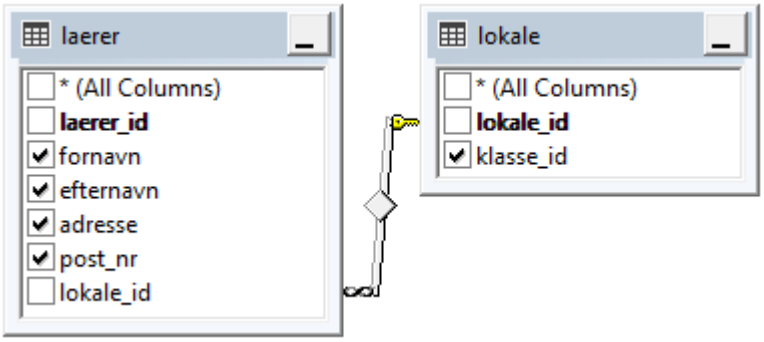
```
select fornavn, efternavn, adresse, elev.post_nr, by_navn
from elev
join post_nr_by
on elev.post_nr=post_nr_by.post_nr
where elev.post_nr = 2500;
```

Den nye tabel kommer til at se således ud:

	fornavn	efternavn	adresse	post_nr	by_navn
1	Jens	Mikkelsen	Lily Brogbergs Vej 17	2500	Valby
2	Henrik	Madsen	Lily Brobergs Vej 53	2500	Valby
3	Rasmus	Nielsen	Nøddehaven 58	2500	Valby

Eksempel 2

```
select fornavn, efternavn, adresse, laerer.post_nr, lokale.klasse_id
from laerer
join lokale
on laerer.lokale_id = lokale.lokale_id;
```

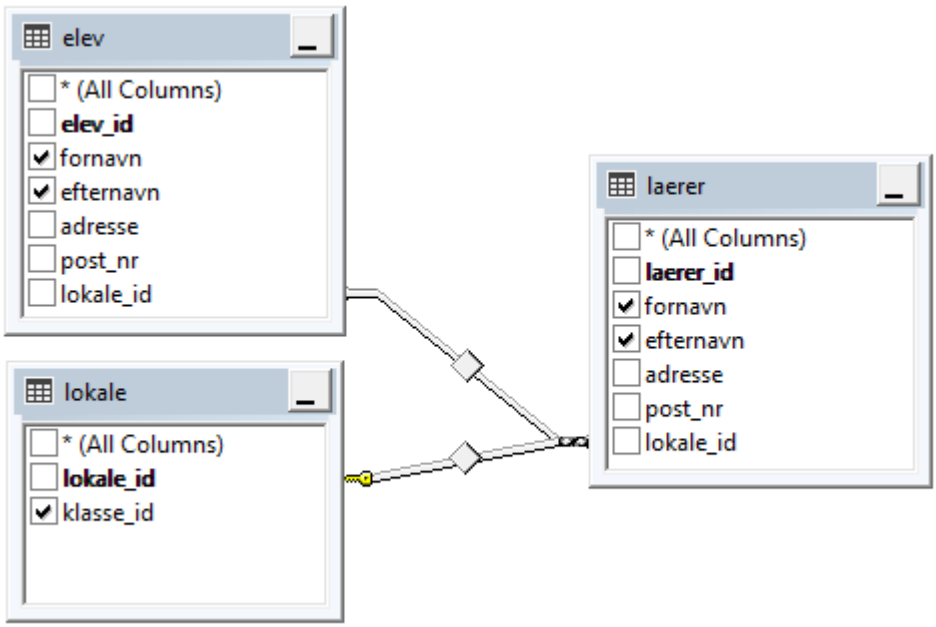


Det ønskes at få vist lærernes navne, postnummer og hvilken klasse de tilhører. Tabellen lokale joines på tabellen laerer på baggrund af lokale_id variablen, som sættes lig hinanden. Den nye tabel vises således:

	fornavn	efternavn	adresse	post_nr	klasse_id
1	Tom	It	Sankt Thomas Alle 3	1824	1A
2	Lars	Henriksen	Nissedalen 76	2740	9B
3	Mia	Hansen	Østervej 16	2750	4D

Eksempel 3

```
select elev.fornavn, elev.efternavn, laerer.fornavn, laerer.efternavn, lokale.klasse_id
from elev
join laerer
on elev.lokale_id = laerer.lokale_id
join lokale
on lokale.lokale_id = laerer.lokale_id;
```



I dette eksempel joines data fra tre forskellige tabeller. Den ønskede tabel skal vise hvilke elever der har hvilken lærer og hvilken klasse de hver især tilhører. Dette gøres ved at lave to joins i samme select. Derved kan man joine flere tabeller på den samme tabel. De linkes alle på baggrund af lokale_id variablen i hver tabel. Den nye tabel ser således ud:

	fornavn	efternavn	fornavn	efternavn	klasse_id
1	Bo	Andersen	Lars	Henriksen	9B
2	Frederikke	Hansen	Tom	It	1A
3	Jens	Mikkelsen	Lars	Henriksen	9B
4	Philip	Mortensen	Mia	Hansen	4D
5	Kasper	Frederiksen	Tom	It	1A
6	Milla	Jørgensen	Mia	Hansen	4D
7	Fie	Knudsen	Lars	Henriksen	9B
8	Henrik	Madsen	Mia	Hansen	4D
9	Rasmus	Nielsen	Tom	It	1A

Programkode

```
CREATE TABLE elev(elev_id int primary key, fornavn TEXT, efternavn TEXT, adresse TEXT, post_nr int, lokale_id
int);
CREATE TABLE post_nr_by (post_nr int primary key, by_navn TEXT);
CREATE TABLE lokale(lokale_id int primary key, klasse_id text);
CREATE TABLE laerer(laerer_id int primary key, fornavn TEXT, efternavn TEXT, adresse TEXT, post_nr INT
foreign key references post_nr_by(post_nr), lokale_id int foreign key references lokale(lokale_id));

alter table elev Add foreign key (post_nr) references post_nr_by(post_nr);
alter table elev Add foreign key (lokale_id) references lokale(lokale_id);

INSERT INTO elev VALUES (1, 'Bo', 'Andersen', 'Gammel Byvej 12', 2650, 2);
INSERT INTO elev VALUES (2, 'Frederikke', 'Hansen', 'Amager Boulevard 5', 2300, 1);
INSERT INTO elev VALUES (3, 'Jens', 'Mikkelsen', 'Lily Brogbergs Vej 17', 2500, 2);
insert into elev values (4, 'Philip', 'Mortensen', 'Brunevang 90', 2610, 3);
insert into elev values (5, 'Kasper', 'Frederiksen', 'Bryggertorvet 32', 3650, 1);
insert into elev values (6, 'Milla', 'Jørgernsen', 'Virum Torv 25', 2830, 3);
insert into elev values (7, 'Fie', 'Knudsen', 'Allen 85', 2770, 2);
insert into elev values (8, 'Henrik', 'Madsen', 'Lily Brobergs Vej 53', 2500, 3);
insert into elev values (9, 'Rasmus', 'Nielsen', 'Nøddehaven 58', 2500, 1);


insert into post_nr_by values (2650, 'Hvidovre');
insert into post_nr_by values (2300, 'København S');
insert into post_nr_by values (2500, 'Valby');
insert into post_nr_by values (2610, 'Rødovre');
insert into post_nr_by values (3650, 'Ølstykke');
insert into post_nr_by values (2830, 'Virum');
insert into post_nr_by values (2770, 'Kastrup');
insert into post_nr_by values (1824, 'Frederiksberg C');
insert into post_nr_by values (2740, 'Skovlunde');
insert into post_nr_by values (2750, 'Ballerup');


insert into lokale values (1, '1A');
insert into lokale values (2, '9B');
insert into lokale values (3, '4D');


insert into laerer values (1, 'Tom', 'It', 'Sankt Thomas Alle 3', 1824, 1);
insert into laerer values (2, 'Lars', 'Henriksen', 'Nissedalen 76', 2740, 2);
insert into laerer values (3, 'Mia', 'Hansen', 'Østervej 16', 2750, 3);


SELECT elev.fornavn, elev.efternavn, lokale.klasse_id
FROM elev, lokale
WHERE elev.lokale_id = lokale.lokale_id ;


select fornavn, efternavn, adresse, elev.post_nr, by_navn
from elev
join post_nr_by
on elev.post_nr=post_nr_by.post_nr
where elev.post_nr = 2500;


select fornavn, efternavn, adresse, laerer.post_nr, lokale.klasse_id
from laerer
join lokale
on laerer.lokale_id = lokale.lokale_id;


select elev.fornavn, elev.efternavn, laerer.fornavn, laerer.efternavn, lokale.klasse_id
from elev
join laerer
on elev.lokale_id = laerer.lokale_id
```

```
join lokale
on lokale.lokale_id = laerer.lokale_id;
```

Konklusion

Udarbejdelse af opgaven har givet mig en grundforståelse for hvordan SQL skrives og fungerer i praksis. Der var nogle forhindringer undervejs, i forhold til installation og i selve koden, som helt sikkert har hjulpet mig til næste gang jeg skal arbejde med relation mellem databaser. Jeg opdagede meget sent i processen at der skulle benyttes snake case, så det blev ikke lige denne gang. Der skulle oprettes et E/R diagram, men min version af SSMS indeholdt ikke den funktion, så prøvede at lave en join hvor alle tabellerne var inkluderet på tværs af hinanden hvor det gav mening, og brugte Query Designer: