



# Databases

# SQL en beheer

Versie 1.4 (20220812)

Niveau 4

IT Systems & Devices

ROC Mondriaan Den Haag

## Inhoud

<b>Inleiding.....</b>	9
<b>Hoofdstuk 1 Tabellen in Excel .....</b>	10
<b>1.1 Rijen, kolommen en velden.....</b>	10
<b>1.2 Kijken naar gegevens in tabellen .....</b>	10
<b>1.3 Selecteren van gegevens uit een tabel in Excel .....</b>	13
<b>Hoofdstuk 2 Gegevens en Informatie .....</b>	15
<b>2.1 Verschil tussen gegevens en informatie .....</b>	15
<b>Hoofdstuk 3 Databases.....</b>	16
<b>3.1 Database .....</b>	16
<b>3.2 Database begrippen.....</b>	17
<b>3.3 Datamodel.....</b>	19
<b>3.4 Tabellen, velden of kolommen, records .....</b>	20
<b>3.5 Primaire sleutel / primairy key.....</b>	20
<b>3.6 Refererende sleutel / foreign key .....</b>	20
<b>3.7 Soorten relaties.....</b>	20
<b>3.8 Database Management System (DBMS).....</b>	22
<b>3.9 Stand-alone DBMS.....</b>	22
<b>3.10 DBMS op server .....</b>	23
<b>3.11 DBMS op dedicated database server .....</b>	24
<b>3.12 Distributed databases.....</b>	24
<b>3.13 Standby databases.....</b>	25
<b>3.14 Complex database beheer .....</b>	26
<b>Hoofdstuk 4 SQL.....</b>	27
<b>4.1 ANSI SQL.....</b>	27
<b>4.2 Geschiedenis van SQL .....</b>	27
<b>4.3 Soorten SQL-opdrachten .....</b>	27
<b>4.4 Drie soorten SQL opdrachten .....</b>	28
<b>4.5 T-SQL .....</b>	28
<b>Hoofdstuk 5 SSMS Graphical User Interface.....</b>	29
<b>5.1 Opstarten van SSMS .....</b>	29
<b>5.2 Databases zichtbaar maken in SSMS .....</b>	30
<b>5.3 Tabellen zichtbaar maken in SSMS .....</b>	31
<b>5.4 Kolommen zichtbaar maken in SSMS .....</b>	31

<b>5.5 Gegevens in tabel zichtbaar maken in SSMS .....</b>	31
<b>5.6 ERD zichtbaar maken in SSMS.....</b>	32
<b>5.7 Database backup maken in SSMS .....</b>	33
<b>5.8 Database verwijderen in SSMS .....</b>	33
<b>5.9 Database uit backup terughalen in SSMS .....</b>	34
<b>5.10 Courante database .....</b>	35
<b>5.11 Query blad in SSMS .....</b>	36
<b>5.12 Client-Server Model (NEW) .....</b>	37
<b>Hoofdstuk 6 Northwind datamodel (NEW) .....</b>	39
<b>6.1 Northwind datamodel .....</b>	39
<b>6.2 Relaties in het Northwind datamodel (NIAM).....</b>	43
<b>6.3 Northwind datamodel in SSMS .....</b>	45
<b>6.4 Factuur gegenereerd uit database .....</b>	47
<b>Hoofdstuk 7 SELECT statement .....</b>	49
<b>7.1 SELECT FROM .....</b>	49
<b>7.2 SELECT FROM WHERE .....</b>	51
<b>7.3 SELECT FROM WHERE AND/OR .....</b>	52
<b>7.4 SELECT FROM WHERE AND/OR ORDER BY .....</b>	53
<b>7.5 SELECT DISTINCT FROM WHERE AND/OR ORDER BY .....</b>	54
<b>7.6 SELECT FROM WHERE NOT AND/OR ORDER BY .....</b>	55
<b>7.7 NULL waarde .....</b>	55
<b>7.8 IS NULL en IS NOT NULL .....</b>	56
<b>7.9 SELECT INTO (NEW) .....</b>	57
<b>Hoofdstuk 8 Aggregatie functies .....</b>	59
<b>8.1 Aggregatie functies .....</b>	59
<b>8.2 COUNT .....</b>	61
<b>8.3 SUM .....</b>	61
<b>8.4 AVG.....</b>	61
<b>8.5 MAX.....</b>	61
<b>8.6 MIN.....</b>	62
<b>8.7 IN .....</b>	62
<b>8.8 BETWEEN.....</b>	62
<b>8.9 LIKE .....</b>	62
<b>8.10 ABS (NEW) .....</b>	63

<b>8.11 UNION (NEW) .....</b>	64
<b>8.12 INTERSECT (NEW) .....</b>	64
<b>8.13 EXCEPT (NEW) .....</b>	65
<b>8.14 UNION ALL, INTERSECT ALL en EXCEPT ALL (NEW) .....</b>	66
<b>8.15 NULL waarden bij UNION, INTERSECT en EXCEPT (NEW) .....</b>	66
<b>8.16 Meer dan 2 UNIONS, INTERSECTS en EXCEPTS (NEW) .....</b>	66
<b>8.17 GROUP BY (NEW) .....</b>	66
<b>8.18 HAVING (NEW) .....</b>	69
<b>8.19 COMMON TABLE EXPRESSION CTE (NEW) .....</b>	69
<b>Hoofdstuk 9 CREATE,INSERT,UPDATE,ALTER,DELETE,DROP, Scripts .....</b>	71
<b>9.1 CREATE DATABASE en USE &lt;databasenaam&gt; .....</b>	71
<b>9.2 CREATE TABLE (NEW) .....</b>	71
<b>9.3 DATE, DATETIME datatype en functie GETDATE (NEW) .....</b>	72
<b>9.4 INSERT INTO .....</b>	73
<b>9.5 UPDATE TABLE .....</b>	74
<b>9.6 ALTER TABLE .....</b>	75
<b>9.7 DELETE FROM .....</b>	75
<b>9.8 TRUNCATE TABLE .....</b>	77
<b>9.9 DROP TABLE .....</b>	77
<b>9.10 DROP DATABASE .....</b>	78
<b>9.11 SQL Scripts (NEW) .....</b>	79
<b>Hoofdstuk 10 NOT NULL, UNIQUE, PRIMARY KEY Constraints .....</b>	83
<b>10.1 Integriteitsregels .....</b>	83
<b>10.2 NOT NULL Constraint .....</b>	84
<b>10.3 UNIQUE KEY Constraint .....</b>	85
<b>10.4 PRIMARY KEY Constraint .....</b>	87
<b>10.5 ALTER TABLE ... ADD PRIMARY KEY Constraint .....</b>	89
<b>10.6 CANDIDATE KEY Constraint en ALTERNATE KEY Constraint .....</b>	89
<b>Hoofdstuk 11 FOREIGN KEY Constraints .....</b>	90
<b>11.1 FOREIGN KEY Constraint .....</b>	90
<b>11.2 ALTER TABLE ... ADD FOREIGN KEY Constraint .....</b>	91
<b>11.3 CREATE TABLE ... FOREIGN KEY .....</b>	91
<b>11.4 ON UPDATE/DELETE action .....</b>	95
<b>11.5 Voorbeeld van toepassing van een Foreign Key Constraint .....</b>	99

<b>11.6 FOREIGN KEYs in het Northwind datamodel .....</b>	100
<b>11.7 FOREIGN KEYs in Entity Relation Diagram .....</b>	101
<b>11.8 Query voor het opvragen van de aanwezige FOREIGN KEY Constraints .....</b>	103
<b>Hoofdstuk 12 CHECK Constraints .....</b>	105
<b>    12.1 CHECK Constraints .....</b>	105
<b>Hoofdstuk 13 SUB-QUERIES .....</b>	107
<b>    13.1 SUB-queries.....</b>	107
<b>Hoofdstuk 14 VIEWS .....</b>	112
<b>    14.1 Views .....</b>	112
<b>Hoofdstuk 15 JOINS .....</b>	115
<b>    15.1 Joins.....</b>	115
<b>    15.2 Impliciete en Expliciete Joins .....</b>	115
<b>    15.3 Pseudoniemen .....</b>	119
<b>    15.4 INNER JOIN .....</b>	121
<b>    15.5 LEFT OUTER JOIN .....</b>	122
<b>    15.6 RIGHT OUTER JOIN.....</b>	124
<b>    15.7 FULL OUTER JOIN .....</b>	124
<b>    15.8 CROSS JOIN .....</b>	125
<b>Hoofdstuk 16 SEQUENCES – (NEW).....</b>	127
<b>    16.1 Sequences .....</b>	127
<b>Hoofdstuk 17 PROGRAMMEERBARE OBJECTEN – (NEW) .....</b>	130
<b>    17.1 Variabelen.....</b>	130
<b>    17.2 Batches .....</b>	130
<b>    17.3 Een batch als een unit of parsing .....</b>	131
<b>    17.4 Batches en variabelen .....</b>	131
<b>    17.5 Statements die niet in een batch gecombineerd kunnen worden .....</b>	132
<b>    17.6 Batch als oplossingsunit .....</b>	132
<b>    17.7 De GO n optie.....</b>	133
<b>    17.8 IF ... ELSE .....</b>	134
<b>    17.9 WHILE .....</b>	135
<b>    17.10 Cursors .....</b>	139
<b>    17.11 Temporary tables.....</b>	142
<b>    17.12 Table Variables .....</b>	143
<b>    17.13 Table types.....</b>	144

<b>17.14 Dynamic SQL .....</b>	144
<b>Hoofdstuk 18 STORED PROCEDURES – (NEW) .....</b>	146
<b>18.1 Stored Procedures .....</b>	146
<b>Hoofdstuk 19 STORED FUNCTIONS - (NEW) .....</b>	150
<b>19.1 Stored Functions .....</b>	150
<b>Hoofdstuk 20 TRIGGERS - (NEW) .....</b>	152
<b>20.1 Triggers.....</b>	152
<b>Hoofdstuk 21 DATABASEBEHEER .....</b>	155
<b>21.1 Taken/verantwoordelijkheden van de databasebeheerder .....</b>	155
<b>Hoofdstuk 22 BACKUP &amp; RECOVERY .....</b>	156
<b>22.1 Backups .....</b>	156
<b>22.1.1 Recovery Models .....</b>	156
<b>22.1.2 Transaction log.....</b>	157
<b>22.1.3 Full Backup, Incremental Backup en Differential backup .....</b>	158
<b>22.1.4 Point in Time Recovery .....</b>	159
<b>22.1.5 SQL Server Full Recovery Model .....</b>	159
<b>22.2 Backup.....</b>	160
<b>22.3 Recovery.....</b>	161
<b>22.4 Tail-log backup.....</b>	162
<b>Hoofdstuk 23 USER ADMINISTRATION – TODO .....</b>	164
<b>23.1 User Administration .....</b>	164
<b>23.2 Logins, Users, Server Roles, Database Roles, Owners, schemas.....</b>	166
<b>23.3 Logins .....</b>	166
<b>23.4 Server Roles .....</b>	167
<b>23.5 Owners .....</b>	167
<b>23.6 Owners en Schemas.....</b>	167
<b>23.7 Logins .....</b>	169
<b>23.8 Server Roles .....</b>	173
<b>23.9 Users.....</b>	176
<b>23.10 Database Roles .....</b>	179
<b>23.11 Schemas .....</b>	183
<b>Hoofdstuk 24 CAPACITY MANAGEMENT .....</b>	185
<b>24.1 Capacity Management.....</b>	185
<b>Hoofdstuk 25 PERFORMANCE TUNING - TODO .....</b>	187

<b>25.1 Denormaliseren .....</b>	188
<b>25.2 Indexen.....</b>	189
<b>25.2.1 Creëren van Indexen .....</b>	192
<b>25.2.2 Execution Plan.....</b>	195
<b>25.3 Optimaliseren van instructies .....</b>	196
<b>25.4 Indexing and Performance (Admin Guide) .....</b>	205
<b>25.4.1 SQL Server Protocols .....</b>	205
<b>25.4.2 Query Processor .....</b>	205
<b>25.4.3 Storage Engine layer.....</b>	210
<b>25.4.4 Performance Monitoring overview .....</b>	210
<b>25.4.5 Tools voor het monitoren van de performance.....</b>	213
<b>25.4.6 Activity Monitor .....</b>	213
<b>25.4.7 Performance Monitor (TODO).....</b>	219
<b>25.4.8 SQL Server Profiler .....</b>	219
<b>25.4.9 SQL Trace.....</b>	221
<b>25.4.10 Extended Events XE .....</b>	222
<b>25.4.11 Dynamic management .....</b>	228
<b>25.4.12 Data Collection .....</b>	228
<b>25.4.13 Query Store (TODO) .....</b>	228
<b>25.4.14 Indexen en onderhoud.....</b>	228
<b>Hoofdstuk 26 TRANSACTIONS EN CONCURRENCY (NEW).....</b>	230
<b>26.1 Transactions.....</b>	230
<b>26.2 Locking .....</b>	237
<b>Hoofdstuk 27 SYSTEM CATALOG VIEWS en SYSTEM STORED PROCEDURES (NEW).....</b>	243
<b>27.1 System Catalog.....</b>	243
<b>27.2 Catalog Views.....</b>	243
<b>27.3 System Stored Procedures .....</b>	244
<b>27.4 Gebruik van Catalog Views en System Stored Procedures .....</b>	245
<b>Hoofdstuk 28 SQL SERVER OP LINUX .....</b>	246
<b>28.1 Verbinding maken met SQL Server database op Linux .....</b>	247
<b>28.2 Wegwijs in SQL Server op Linux .....</b>	251
<b>Bijlage 1 Syntax Diagrams .....</b>	252
<b>Bijlage 2 Entity Relation Diagram van Northwind database .....</b>	255
<b>Bijlage 3 SQL script voor genereren Northwind Trades Factuur .....</b>	256



## Inleiding

Deze reader bevat de leerstof voor het vak databases van de opleiding IT Systems & Devices binnen het ROC Mondriaan Den Haag.

In tegenstelling tot de meeste database schoolboeken richt deze reader zich niet op het ontwerpen van databases maar richt het zich op het beheer van de databases.

De databasebeheerder ofwel Database Administrator (DBA) is verantwoordelijk voor het beheer van de databases. Dat betekent dat de databasebeheerder ervoor zorgt dat er nooit data uit de database verloren zal gaan (ook niet bij storingen), dat alle gebruikers van de database de juiste privileges hebben op de data (niet meer en niet minder dan noodzakelijk), dat de databasegroei onder controle is en dat de database altijd snel blijft reageren.



Na de inleidende hoofdstukken 1 t/m 5 over databases in het algemeen wordt in hoofdstuk 6 kennis gemaakt met de Graphical User Interface (GUI) van SQL Server Management Studio (SSMS).

In hoofdstukken 7 t/m 19 wordt de database taal SQL behandeld. Doel van deze hoofdstukken is dat we de SQL gereedschappen leren kennen, zodat we bij het oplossen van problemen de juiste gereedschappen kunnen gebruiken. In deze reader wordt achterliggende theorieën zo veel mogelijk achterwege gelaten en wordt zo veel mogelijk met voorbeelden de werking van de SQL gereedschappen behandeld. De bedoeling is dat de studenten de voorbeelden proberen te begrijpen, zelf proberen na te spelen en in de oefeningen de gereedschappen proberen te gebruiken in vergelijkbare opgaven. Bij het oplossen van problemen en vraagstukken is de inhoud van deze reader waarschijnlijk onvoldoende en zal op de online documentatie sites meer details over de gereedschappen moeten worden opgezocht.

Vanaf hoofdstuk 20 wordt het beheren van databases, zoals Backup & Recovery, User Administration, Capacity Management en Performance Tuning behandeld.

De leerstof over de databasetaal SQL zal behandeld worden in het eerste en in het tweede leerjaar van de opleiding IT Systems & Devices. In het derde leerjaar zal SQL gebruikt worden in complexere opgaven ter voorbereiding op het niveau 4 examen.

De leerstof over het beheer van de databases zal behandeld worden in het tweede en derde leerjaar.

## Hoofdstuk 1 Tabellen in Excel

Voordat we naar databases gaan kijken, kijken we nu eerst naar gegevens in tabellen in Excel.

### 1.1 Rijen, kolommen en velden

Een tabel is een lijst met gegevens op een overzichtelijke manier gerangschikt in een patroon van regelmatige horizontale en verticale lijnen. Een horizontale lijn noemen we een **rij**, een verticale lijn noemen we een **kolom** en het snijpunt van een rij en een kolom noemen we een **veld**.

Tabel



### 1.2 Kijken naar gegevens in tabellen

In de onderstaande tabel staan gegevens van studenten en gegevens van de klassen waarin ze zitten:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Studentnummer	Voorletters	Achternaam	Geslacht	geboortedatum	Adres	Huisnummer	Postcode	Woonplaats	telefoonnummer	Klas	Locatie	Adres	Postcode	Plaats	Klassedocent
2	S0001	A.	Baan	M	12-5-2004	Terwestenstraat	18	2525GH	Den Haag	06-12345678	1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P. Jansen
3	S0002	S.	Hijmans	V	7-12-2004	Sweelinckplein	16	2517GM	Den Haag	06-23456789	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
4	S0003	J.	Jansen	M	14-3-2004	Brinkershof	17	2713TX	Zoetermeer	071-1234567	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
5	S0004	J.	Jansen	M	14-3-2004	Brinkershof	17	2713TX	Zoetermeer	071-1234567	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
6	S0005	A.	Maarsen	M	12-5-2004	Rembrandtstraat	2	2526PZ	Den Haag	06-56789012	1E	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
7	S0006	K.	Jakobs	V	12-12-2005	De Vliegerstraat	55	2525BG	Den Haag	06-67890123	1F	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P. Jansen
8	S0007	A.B.	Willemsen	M	14-5-2004	Terwestenstraat	16	2525GH	Den Haag	06-78901234	1G	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
9	S0008	U.	Jans	M	17-6-2005	Herman Costerstraat	162	2571PD	Den Haag	06-89012345	1H	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
10	S0009	S.	Huber	M	12-5-2004	Hobbemastraat	22	2526JP	Den Haag	06-90123456	1I	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P. Jansen
11	S0010	Z.	Wessels	V	5-8-2004	Retiefstraat	29	2571PS	Den Haag	06-13579135	1J	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
12	S0011	A.	Kassenberg	M	12-5-2006	Almeloplein	12	2533AA	Den Haag	06-24680246	1K	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
13	S0012	L.	Vrogers	M	12-5-2004	van der Vennestraat	85	2525CC	Den Haag	06-35791357	1L	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
14	S0013	A.D.	Elfering	M	5-9-2005	Reitstraat	52	2571RT	Den Haag	06-57913579	1M	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P. Jansen
15	S0014	I.	Sebastiaans	M	12-5-2004	Rembrandtstraat	1	2526PN	Den Haag	06-79135791	1N	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
16	S0015	J.	de Jong	M	12-12-2006	van Miereveldstraat	62	2525EG	Den Haag	06-91357913	1O	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
17	S0016	T.	Boeren	V	12-8-2004	Herman Costerstraat	164	2571PD	Den Haag	06-13456789	1P	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
18	S0017	J.	Huizinga	M	12-5-2005	Terwestenstraat	20	2525GH	Den Haag	06-25678901	1Q	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
19	S0018	D.	Pol	M	12-12-2004	Wesselsstraat	283	2572RZ	Den Haag	06-47890123	1R	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P. Jansen
20	S0019	E.	Reitsema	M	12-5-2004	Retiefstraat	27	2571PS	Den Haag	06-47890123	1S	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P. Jansen
21	S0020	P.	Vos	M	5-3-2004	van Miereveldstraat	64	2525EG	Den Haag	06-69012345	1T	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
22	S0021	K.	Hazenberg	V	12-5-2004	van der Vennestraat	83	2525CC	Den Haag	06-71234567	1U	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
23	S0022	L.	Joemann	M	12-5-2005	Hobbemastraat	20	2526JP	Den Haag	06-94567890	1V	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P. Jansen
24	S0023	S.	Claasen	M	8-9-2004	Terwestenstraat	111	2525GG	Den Haag	06-10987654	1W	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
25	S0024	W.	de Wit	M	12-5-2004	Almeloplein	10	2533AA	Den Haag	06-29876543	1X	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
26	S0025	O.	Snellers	M	12-10-2005	Herman Costerstraat	160	2571PD	Den Haag	06-30886543	1Y	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P. Jansen
27	S0026	H.	Blokhuis	V	8-10-2006	Rembrandtstraat	3	2526PN	Den Haag	06-41234098	1Z	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
28	S0027	D.	Haverkamp	V	12-5-2004	Retiefstraat	25	2571PS	Den Haag	06-52345679	2A	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
29	S0028	B.	Bol	M	12-12-2006	Terwestenstraat	22	2525GH	Den Haag	06-59873456	2B	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
30	S0027	K.	Reijvenkamp	M	12-12-2006	Terwestenstraat	50	2571RT	Den Haag	06-59873112	2C	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
31	S0028	L.	van den Broek	M	12-5-2004	Wesselsstraat	281	2572RZ	Den Haag	06-22093586	2D	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen

Links van de kolom **Klas** staan de gegevens van de studenten (naam, adres, etc.) en rechts van de kolom **Klas** staan de gegevens van de locatie van de klassen (gebouw, adres, etc.). Er staan dus twee groepen gegevens in dezelfde tabel. Gegevens over de studenten en gegevens over de klassen.

Doordat beide groepen gegevens in de tabel staan zien we dat er in het rechterdeel van de tabel veel dubbele gegevens staan. Meerdere studenten zitten in klas 1A, meerdere studenten zitten in klas 1B, etc. Bij elke student staat vermeld wat het adres van de locatie van zijn/haar klas is. Er staat dus meerdere malen het adres van klas 1A en meerdere malen het adres van klas 1B, etc. in de tabel opgeslagen.

We zouden nu de tabel kunnen opsplitsen in twee kleinere tabellen waarbij er een koppeling tussen de twee tabellen bestaat op de kolom **klas**. Elke student zit immers in een klas en elke klas zit op een bepaalde locatie.

A	B	C	D	E	F	G	H	I	J	K
Studentnummer	Voorletters	Achternaam	Geslacht	geboortedatum	Adres	Huisnummer	Postcode	Woonplaats	telefoonnummer	Klas
1	S0001	A.	Baan	M	12-5-2004 Terwestenstraat	18	2525GH	Den Haag	06-12345678	1A
2	S0002	S.	Hijmans	V	7-12-2004 Sweelinckplein	16	2517GM	Den Haag	06-23456789	1B
3	S0003	J.	Jansen	M	14-3-2004 Brinkershof	17	2713TX	Zoetermeer	071-1234567	1C
4	S0004	J.	Jansen	M	14-3-2004 Brinkershof	17	2713TX	Zoetermeer	071-1234567	1C
5	S0005	A.	Maarsen	M	12-5-2004 Rembrandtstraat	2	2526PZ	Den Haag	06-56789012	1D
6	S0006	K.	Jakobs	V	12-12-2005 De Vliegerstraat	55	2525BG	Den Haag	06-67890123	1B
7	S0007	A.B.	Willemsen	M	14-5-2004 Terwestenstraat	16	2525GH	Den Haag	06-78901234	1B
8	S0008	U.	Jan	M	17-6-2005 Herman Costerstraat	162	2571PD	Den Haag	06-89012345	1C
9	S0009	S.	Huber	M	12-5-2004 Hobbeismastraat	22	2526P	Den Haag	06-90123456	1A
10	S0010	Z.	Wessels	V	5-8-2004 Reitiefstraat	29	2571PS	Den Haag	06-13579135	1B
11	S0011	A.	Kassenberg	M	12-5-2006 Almeloplein	12	2533AA	Den Haag	06-24680246	1D
12	S0012	L.	Vrogers	M	12-5-2004 van der Vennenstraat	85	2525CC	Den Haag	06-35791357	1C
13	S0013	A.D.	Elfering	M	5-9-2005 Reitzastraat	52	2571RT	Den Haag	06-57913579	1A
14	S0014	I.	Sebastiaans	M	12-5-2004 Rembrandtstraat	1	2526PN	Den Haag	06-79135791	1C
15	S0015	J.	de Jong	M	12-12-2006 van Miereveldstraat	62	2525EG	Den Haag	06-91357913	1D
16	S0016	T.	Boeren	V	12-8-2004 Herman Costerstraat	164	2571PD	Den Haag	06-14567899	1B
17	S0017	J.	Huizinga	M	12-5-2005 Terwestenstraat	20	2525GH	Den Haag	06-25678901	1D
18	S0018	D.	Pol	M	12-12-2004 Wesselstraat	283	2572RZ	Den Haag	06-7890123	1A
19	S0019	E.	Reitsema	M	12-5-2004 Reitiefstraat	27	2571PS	Den Haag	06-47890123	1A
20	S0020	P.	Vos	M	5-3-2004 van Miereveldstraat	64	2571PN	Den Haag	06-69012345	1B
21	S0021	K.	Hazenberg	V	12-5-2004 van der Vennenstraat	83	2525CC	Den Haag	06-71234567	1D
22	S0022	L.	Joemann	M	12-5-2005 Hobbeismastraat	20	2526P	Den Haag	06-94567890	1A
23	S0023	S.	Clasen	-	8-9-2004 Terwestenstraat	111	2525GG	Den Haag	06-10987654	1C
24	S0024	W.	de Wit	M	12-5-2004 Almeloplein	10	2533AA	Den Haag	06-29876543	1D
25	S0025	O.	Snellers	M	12-10-2005 Herman Costerstraat	160	2571PD	Den Haag	06-30865543	1A
26	S0026	H.	Blokhus	V	8-10-2006 Rembrandtstraat	3	2526PN	Den Haag	06-41234098	1D
27	S0027	D.	Haverkamp	V	12-5-2004 Reitiefstraat	25	2571PS	Den Haag	06-52345679	1B
28	S0028	B.	Bol	M	12-12-2006 Terwestenstraat	22	2525GH	Den Haag	06-59873456	1D
29	S0029	K.	Reuvenkamp	M	22-11-2005 Reitzastraat	50	2571RT	Den Haag	06-59873112	1D
30	S0030	L.	van den Broek	M	12-5-2004 Wesselstraat	281	2572RZ	Den Haag	06-22095586	1C

A	B	C	D	E	F
Klas	Locatie	Adres	Postcode	Plaats	Klassedocent
1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P. Jansen
3B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
4C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
5C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
6D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
7B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
8B	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P. Jansen
9C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
10A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P. Jansen
11B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
12D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
13C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
14A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P. Jansen
15C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
16D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
17B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
18D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
19A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P. Jansen
20A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P. Jansen
21B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
22D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
23A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P. Jansen
24C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
25D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
26A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P. Jansen
27D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
28B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
29D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
30D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
31C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen

In de tabel links staan nu de gegevens van de studenten en in de tabel rechts staan nu de gegevens van de klassen. In de rechertabel zien we nu dat elke locatie (gebouw) meerdere malen in de tabel staan met dezelfde adres gegevens. Dat noemen we **redundante** (overtollige) gegevens. Het zou voldoende moeten zijn als deze gegevens per locatie slechts één maal in de tabel zou staan.

We kunnen nu de dubbele gegevens in de rechter tabel weghalen zodat er een kleinere tabel over blijft.

A	B	C	D	E	F	G	H	I	J	K
Studentnummer	Voorletters	Achternaam	Geslacht	geboortedatum	Adres	Huisnummer	Postcode	Woonplaats	telefoonnummer	Klas
1	S0001	A.	Baan	M	12-5-2004 Terwestenstraat	18	2525GH	Den Haag	06-12345678	1A
2	S0002	S.	Hijmans	V	7-12-2004 Sweelinckplein	16	2517GM	Den Haag	06-23456789	1B
3	S0003	J.	Jansen	M	14-3-2004 Brinkershof	17	2713TX	Zoetermeer	071-1234567	1C
4	S0004	J.	Jansen	M	14-3-2004 Brinkershof	17	2713TX	Zoetermeer	071-1234567	1C
5	S0005	A.	Maarsen	M	12-5-2004 Rembrandtstraat	2	2526PZ	Den Haag	06-56789012	1D
6	S0006	K.	Jakobs	V	12-12-2005 De Vliegerstraat	55	2525BG	Den Haag	06-67890123	1B
7	S0007	A.B.	Willemsen	M	14-5-2004 Terwestenstraat	16	2525GH	Den Haag	06-78901234	1B
8	S0008	U.	Jans	M	17-6-2005 Herman Costerstraat	162	2571PD	Den Haag	06-89012345	1C
9	S0009	S.	Huber	M	12-5-2004 Hobbeismastraat	22	2526P	Den Haag	06-90123456	1A
10	S0010	Z.	Wessels	V	5-8-2004 Reitiefstraat	29	2571PS	Den Haag	06-13579135	1B
11	S0011	A.	Kassenberg	M	12-5-2006 Almeloplein	12	2533AA	Den Haag	06-24680246	1D
12	S0012	L.	Vrogers	M	12-5-2004 van der Vennenstraat	85	2525CC	Den Haag	06-35791357	1C
13	S0013	A.D.	Elfering	M	5-9-2005 Reitzastraat	52	2571RT	Den Haag	06-57913579	1A
14	S0014	I.	Sebastiaans	M	12-5-2004 Rembrandtstraat	1	2526PN	Den Haag	06-79135791	1C
15	S0015	J.	de Jong	M	12-12-2006 van Miereveldstraat	62	2525EG	Den Haag	06-91357913	1D
16	S0016	T.	Boeren	V	12-8-2004 Herman Costerstraat	164	2571PD	Den Haag	06-14567899	1B
17	S0017	J.	Huizinga	M	12-5-2005 Terwestenstraat	20	2525GH	Den Haag	06-25678901	1D
18	S0018	D.	Pol	M	12-12-2004 Wesselstraat	283	2572RZ	Den Haag	06-7890123	1A
19	S0019	E.	Reitsema	M	12-5-2004 Reitiefstraat	27	2571PS	Den Haag	06-47890123	1A
20	S0020	P.	Vos	M	5-3-2004 van Miereveldstraat	64	2525EG	Den Haag	06-69012345	1B
21	S0021	K.	Hazenberg	V	12-5-2004 van der Vennenstraat	83	2525CC	Den Haag	06-71234567	1D
22	S0022	L.	Joemann	M	12-5-2005 Hobbeismastraat	20	2526P	Den Haag	06-94567890	1A
23	S0023	S.	Clasen	-	8-9-2004 Terwestenstraat	111	2525GG	Den Haag	06-10987654	1C
24	S0024	W.	de Wit	M	12-5-2004 Almeloplein	10	2533AA	Den Haag	06-29876543	1D
25	S0025	O.	Snellers	M	12-10-2005 Herman Costerstraat	160	2571PD	Den Haag	06-30986543	1A
26	S0026	H.	Blokhus	V	8-10-2006 Rembrandtstraat	3	2526PN	Den Haag	06-41234098	1D
27	S0027	D.	Haverkamp	V	12-5-2004 Reitiefstraat	25	2571PS	Den Haag	06-52345679	1B
28	S0028	B.	Bol	M	12-12-2006 Terwestenstraat	22	2525GH	Den Haag	06-59873456	1D
29	S0029	K.	Reuvenkamp	M	22-11-2005 Reitzastraat	50	2571RT	Den Haag	06-59873112	1D
30	S0030	L.	van den Broek	M	12-5-2004 Wesselstraat	281	2572RZ	Den Haag	06-22095586	1C

A	B	C	D	E	F
Klas	Locatie	Adres	Postcode	Plaats	Klassedocent
2 1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P. Jansen
3 1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
4 1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
5 1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten

In de linker tabel kunnen we vinden in welke klas een student zit en in de rechter tabel kunnen we dan vinden op welke locatie deze klas zich bevindt. Via de kolom klas wordt er vanuit de linker tabel gerefereerd naar de kolom Klas in de rechter tabel. Deze **referentie** (blauwe lijn) wordt een **Foreign Key constraint** (verwijzende sleutel beperking) genoemd en zullen we in de database hoofdstukken verder behandelen.

De voordelen van de opsplitsing van de grote tabel en het verwijderen van redundante gegevens zijn:

- Elke tabel bevat slechts één gegevensgroep, dus eenduidig
- Zoeken in een kleine tabel is sneller dan zoeken in een grote tabel
- Kleine tabel neemt minder diskruimte in beslag dan een grote tabel
- Minder gegevens in kleine tabel, dus ook minder kans op fouten

Het opsplitsen van gegevensgroepen in aparte tabellen wordt **normaliseren** genoemd. Normaliseren maakt deel uit van het ontwerpen van databases en wordt verder niet behandeld in deze reader.

Als we goed naar de gegevens in de oorspronkelijke grote tabel kijken dan zien we dat er veel twijfelachtige gegevens in de tabel staan:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Studentnummer	Voorletters	Achternaam	Geslacht	Geboortedatum	Adres	Huisnummer	Postcode	Woonplaats	telefoonnummer	Klas	Locatie	Adres	Postcode	Plaats	Klassedocent
2	S0001	A.	Baan	M	12-5-2004	Terwestenstraat	18	2525GH	Den Haag	06-12345678	1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P.Jansen
3	S0002	S.	Hijmans	V	7-12-2004	Sweelinkplein	16	2517GM	Den Haag	06-23456789	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
4	S0003	J.	Jansen	M	14-3-2004	Brinkershof	17	2713TX	Zoetermeer	071-1234567	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
5	S0004	J.	Jansen	M	14-3-2004	Brinkershof	17	2713TX	Zoetermeer	071-1234567	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
6	S0005	A.	Maarsen	M	12-5-2004	Rembrandtstraat	2	2526PZ	Den Haag	06-56789012	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
7	S0006	K.	Jakobs	V	12-12-2005	De Vliegerstraat	55	2525BG	Den Haag	06-67890123	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
8	S0007	A.B.	Willemsen	M	14-5-2004	Terwestenstraat	16	2525GH	Den Haag	06-78901234	1B	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P.Jansen
9	S0008	U.	Jans	M	17-6-2005	Herman Costerstraat	162	2571PD	Den Haag	06-89012345	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
10	S0009	S.	Huber	M	12-5-2004	Hobbeismastraat	22	2526JP	Den Haag	06-90123456	1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P.Jansen
11	S0010	Z.	Wessels	V	5-8-2004	Retiefstraat	29	2571PS	Den Haag	06-13579135	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
12	S0011	A.	Kassenberg	M	12-5-2006	Almeloplein	12	2533AA	Den Haag	06-24680246	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
13	S0012	L.	Vrogers	M	12-5-2004	van der Vennestraat	85	2525CC	Den Haag	06-35791357	1A	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
14	S0013	A.D.	Elfering	M	5-9-2005	Retiefstraat	52	2571RT	Den Haag	06-57913579	1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P.Jansen
15	S0014	I.	Sebastiaans	M	12-5-2004	Rembrandtstraat	1	2526PN	Den Haag	06-79135791	1A	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
16	S0015	J.	de Jong	M	12-12-2006	van Miereveldstraat	62	2525EG	Den Haag	06-91357913	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
17	S0016	T.	Boeren	V	12-8-2004	Herman Costerstraat	164	2571PD	Den Haag	06-123456789	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
18	S0017	J.	Huizinga	M	12-5-2005	Terwestenstraat	20	2525GH	Den Haag	06-25678901	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
19	S0018	D.	Pol	M	12-12-2004	Wesselsstraat	283	2572RZ	Den Haag		1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P.Jansen
20	S0019	E.	Reitsema	M	12-5-2004	Retiefstraat	27	2571PS	Den Haag	06-47890123	1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P.Jansen
21	S0020	P.	Vos	M	5-3-2004	van Miereveldstraat	64	2525EG	Den Haag	06-69012345	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
22	S0021	K.	Hazenberg	V	12-5-2004	van der Vennestraat	83	2525CC	Den Haag	06-71234567	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
23	S0022	L.	Joemman	M	12-5-2005	Hobbeismastraat	20	2526JP	Den Haag	06-94567890	1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P.Jansen
24	S0023	S.	Claasen	-	8-9-2004	Terwestenstraat	111	2525GG	Den Haag	06-10987654	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
25	S0024	W.	de Wit	M	12-5-2004	Almeloplein	10	2533AA	Den Haag	06-29876543	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
26	S0025	O.	Snellers	M	12-10-2005	Herman Costerstraat	160	2571PD	Den Haag	06-30986543	1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P.Jansen
27	S0026	H.	Blokhuis	V	8-10-2006	Rembrandtstraat	3	2526PN	Den Haag	06-41234098	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
28	S0026	D.	Haverkamp	V	12-5-2004	Retiefstraat	25	2571PS	Den Haag	06-52345679	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
29	S0026	B.	Bol	M	12-12-2006	AAAAAAAAAAAA	22	2525GH	Den Haag	06-59873456	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
30	S0027	K.	Reuvenkamp	M	22-11-2005	Retiefstraat	50	2571RT	Den Haag	06-59873112	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
31	S0028	L.	van den Broek	M	12-5-2004	Wesselsstraat	281	2572RZ	Den Haag	06-22095586	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen

- In de rijen 4 en 5 zien we dat er 2x dezelfde informatie in de tabel staat. Het zou kunnen zijn dat per abuis een student 2x is ingevoerd met een ander studentnummer, maar het zou ook kunnen zijn dat het een tweeling betreft Jan Jansen en Jacob Jansen met hetzelfde adres en dezelfde geboortedatum.
- In de rijen 7 en 8 zien we dat er voor klas 1B twee verschillende locaties staan. Daar een klas slechts op één locatie kan zijn spreken de gegevens elkaar tegen. Dit noemen we **inconsistentie** (tegenstrijdigheid) van de gegevens. Door deze inconsistentie wordt de **integriteit** (betrouwbaarheid) van de tabel of database verminderd.
- In rij 19 zien we dat er geen telefoonnummer ingevuld is. Het kan zijn dat bij de invoer het telefoonnummer vergeten is maar het is ook mogelijk dat de student daadwerkelijk geen telefoon heeft of niet wil dat de school zijn telefoonnummer weet.
- In rij 24 zien we dat er voor het geslacht een - teken is ingevuld. Het kan zijn dat het een type fout was of dat het geslacht van de student bij het invoeren onbekend was, maar het zou ook kunnen zijn dat het een non-binair student betreft.
- In rij 27 zien we een student zonder studentnummer.
- In rij 29 zien we dat de student hetzelfde studentnummer heeft als de student in rij 28.
- In rij 29 zien we tevens dat er een adres AAAAAAAAAAAAAA staat.

Voor al deze bovenstaande twijfelachtige gegevens geldt dat de databasebeheerder zelf geen conclusies over de juistheid van de gegevens kan trekken, maar daarover zal moeten overleggen met de datadeskundigen van de betreffende afdeling.

Waar het hier nu om gaat is dat we zien dat er in Excel niets is dat voorkomt dat er twijfelachtige gegevens in de tabel worden ingevuld. Bij databases kunnen we **constraints** (beperkingen) gebruiken om de invoer van twijfelachtige gegevens te voorkomen.

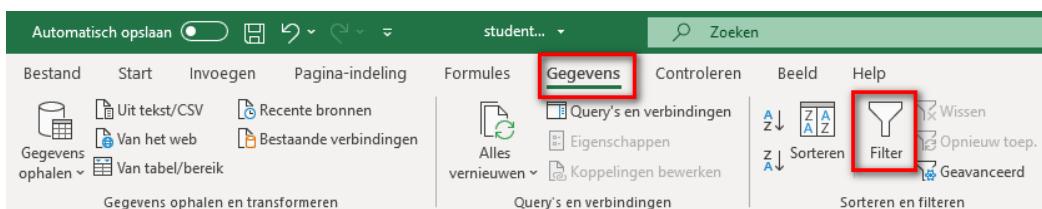
### 1.3 Selecteren van gegevens uit een tabel in Excel

In Excel kunnen we de gewenste gegevens uit de tabel **selecteren** en **filteren**:

- Klik op  links van kolom A en boven rij 1 om alle velden te selecteren

	A	B	C
1	Naam	Leeftijd	Woonplaats
2	Jan	16	Den Haag
3	Raoul	17	Rijswijk
4	Mohammad	18	Den Haag
5	Timothy	17	Naaldwijk

- Klik vervolgens op **Gegevens** en dan op **Filter**



Klik op **pull down menu** knop  van kolom Leeftijd

	A	B	C
1	Naam	Leeftijd	Woonplaats
2	Jan	16	Den Haag
3	Raoul	17	Rijswijk
4	Mohammad	18	Den Haag
5	Timothy	17	Naaldwijk

- Selecteer **17** om alle studenten die 17 jaar zijn te zien en klik op **OK**



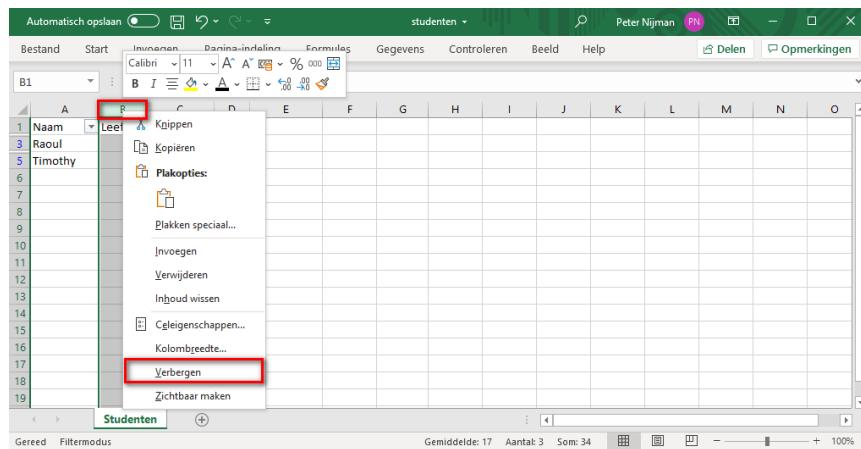
- Merk op dat het **resultaat** nu alleen alle studenten die 17 jaar zijn bevat. Het resultaat geeft mij nu informatie over alle studenten die 17 jaar zijn.

	A	B	C
1	Naam	Leeftijd	Woonplaats
3	Raoul	17	Rijswijk
5	Timothy	17	Naaldwijk

Aan het **filter** teken in de pull down knop  is te zien dat slechts een deel van de gegevens is

geselecteerd (gefilterd).

- Klik met de rechtermuisknop op de kolom B (Leeftijd) en klik op Verbergen



- De kolom Leeftijd is nu uit het overzicht verdwenen

	A	C	D	E
1	Naam	Woonplaats		
3	Raoul	Rijswijk		
5	Timothy	Naaldwijk		
6				

Belangrijk is dat we ons realiseren dat we met bovenstaande stappen in Excel slechts een deel van de gegevens zichtbaar hebben gemaakt. De niet zichtbare gegevens staan niet in het resultaat maar staan nog wel steeds in de tabel!

In databases kunnen we ook de gewenste gegevens selecteren en filteren m.b.v. SELECT FROM WHERE statements. Dit zullen we in de SQL hoofdstukken verder behandelen. Voor nu is het belangrijk om te realiseren dat we de gewenste gegevens uit een database kunnen halen en in het resultaat kunnen weergeven maar de gegevens in de tabel zelf ongewijzigd blijven.

## Hoofdstuk 2 Gegevens en Informatie

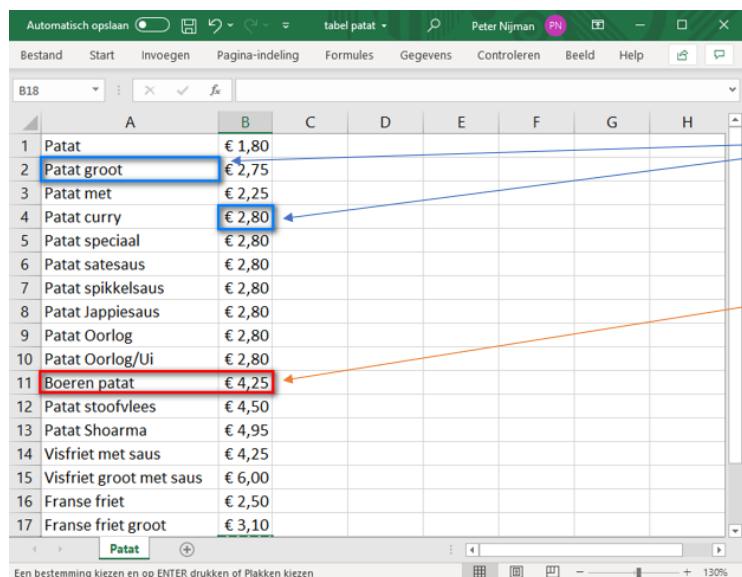
### 2.1 Verschil tussen gegevens en informatie

In de informatica maken we onderscheid tussen gegevens en informatie.

Onder **gegevens (data)** verstaan we losstaande feiten.

Onder **informatie (information)** verstaan we gegevens die door rangschikking een nuttige betekenis geven.

Hieronder volgt een voorbeeld:



	A	B	C	D	E	F	G	H
1	Patat	€ 1,80						
2	Patat groot	€ 2,75						
3	Patat met	€ 2,25						
4	Patat curry	€ 2,80						
5	Patat speciaal	€ 2,80						
6	Patat satesaus	€ 2,80						
7	Patat spikkelsaus	€ 2,80						
8	Patat Jappiesaus	€ 2,80						
9	Patat Oorlog	€ 2,80						
10	Patat Oorlog/Ui	€ 2,80						
11	Boeren patat	€ 4,25						
12	Patat stoofvlees	€ 4,50						
13	Patat Shoarma	€ 4,95						
14	Visfriet met saus	€ 4,25						
15	Visfriet groot met saus	€ 6,00						
16	Franse friet	€ 2,50						
17	Franse friet groot	€ 3,10						

De **gegevens** (losstaande feiten) in de tabel geven door de rangschikking **informatie** (nuttige betekenis) over de prijzen van patat

Wanneer we tegen iemand zeggen ‘Patat groot’ dan vraagt de ander zich af wat er met Patat groot aan de hand is.

Wanneer we tegen iemand zeggen ‘€ 2,80’, dan vraagt de ander zich af wat er € 2,80 kost.

Wanneer we tegen iemand zeggen ‘Boeren patat € 4,25’ dan begrijpt de ander dat Boeren patat € 4,25 kost.

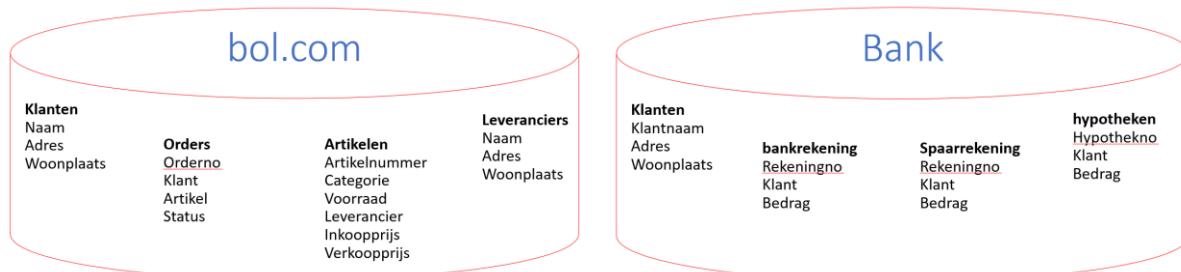
In de eerste twee voorbeelden betreft hetgeen we zeggen slechts losse gegevens die de ander geen nuttige betekenis geven. In het derde voorbeeld betreft hetgeen we zeggen informatie omdat de rangschikking van de gegevens nu aan de ander wel een nuttige betekenis geeft.

In een tabel zit vaak een grote hoeveelheid gegevens. We zijn meestal niet geïnteresseerd in alle gegevens uit de tabel maar slechts in enkele gegevens op een bepaalde manier gerangschikt. Om de juiste gegevens in de juiste rangschikking te verkrijgen moeten we de gegevens op de juiste manier selecteren en filteren. Het zoeken naar de verkeerde gegevens of een verkeerde rangschikking geeft geen foutmelding maar wel een onjuist resultaat!

## Hoofdstuk 3 Databases

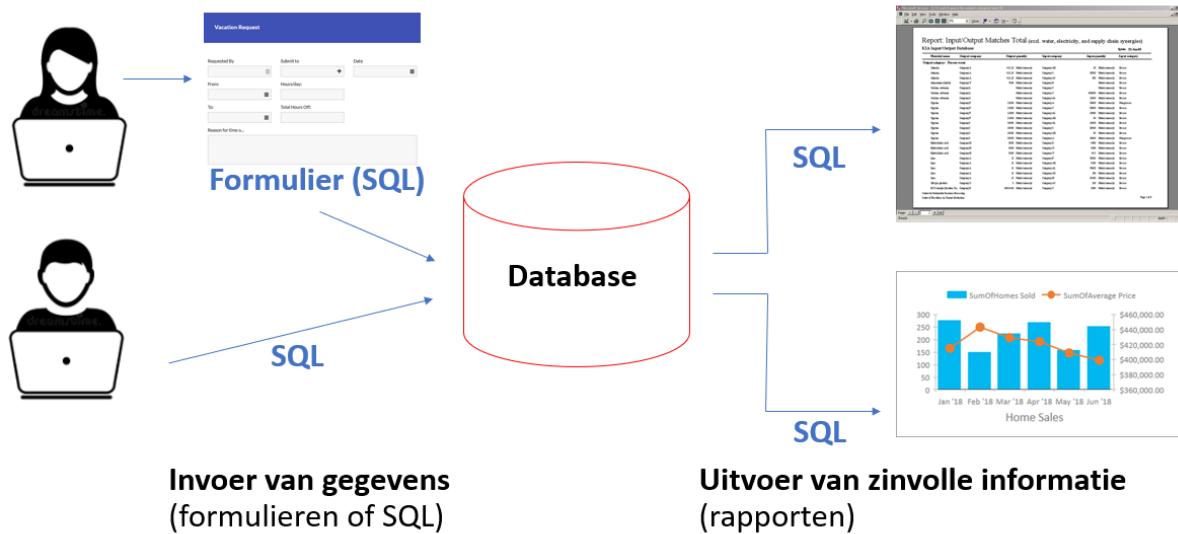
### 3.1 Database

Een **database** kunnen we zien als een grote bak met tabellen. Elke database heeft zijn eigen tabellen en in de tabellen zitten de gegevens van het bedrijf. De gegevens in de tabellen zijn meestal gegevens waar het bedrijf in het primaire proces van afhankelijk is. De tabellen en de gegevens in de tabellen verschillen dus per bedrijf.



De tabellen in een database van een online winkel zoals bol.com bevatten onder andere de gegevens van de klanten, de orders (bestellingen) die de klanten hebben gemaakt, welke artikelen er in de orders zijn besteld, hoeveel de voorraad van de artikelen in het magazijn is en welke leveranciers de artikelen leveren en de gegevens van de leveranciers.

De tabellen in een database van een bank bevatten onder andere de gegevens van de klanten, hoeveel geld elke klant op de bank- of spaarrekening heeft staan en hoeveel geld elke klant heeft geleend.



De gegevens in de tabellen worden veelal door gebruikers m.b.v. een (online) **formulier** ingevoerd. Door het formulier hoeft de gebruiker geen kennis te hebben van de databasetaal SQL en kan de gebruiker de gegevens direct in velden van het formulier invoeren. Zodra de gebruiker op de knop drukt om de gegevens te bevestigen worden de gegevens door het formulier in SQL code naar de database gestuurd en in de database opgeslagen.

Daarnaast worden gegevens ook door applicatiebeheerders of databasebeheerders m.b.v. de databasetaal SQL rechtstreeks in de database ingevoerd. Dit laatste kunnen handmatige invoeren zijn maar kunnen ook **batches** (massa invoer) zijn waarin in één keer grote hoeveelheden data ingevoerd worden.

De gegevens kunnen uit een database worden gehaald m.b.v. SQL commando's. Veelal wordt het resultaat van de geselecteerde gegevens gepresenteerd in een **rappoort**, waarmee verantwoordelijke medewerkers van het bedrijf beslissingen kunnen nemen.

Gegevens in een database zijn niet willekeurig maar **gestructureerd** opgeslagen zodat de gegevens snel teruggevonden kunnen worden. Dit is te vergelijken met een niet gestructureerd opgeslagen verzameling boeken en een wel gestructureerd opgeslagen verzameling boeken.

Niet gestructureerde opslag



Gestructureerde opslag



In een niet gestructureerde opslag (links) is het vrijwel niet mogelijk om snel het juiste boek te selecteren. In een gestructureerde opslag (rechts) is het wél mogelijk om middels een **index** direct naar de juiste kast te lopen en het juiste boek te selecteren. Op zelfde wijze zijn de gegevens in de tabellen gestructureerd opgeslagen en kan middels indexen direct naar de juiste plaats in een tabel worden gegaan om daar de gegevens snel uit te halen.

### 3.2 Database begrippen

In de informatica worden verschillende database begrippen gebruikt:

- **Database** bestaat uit een verzameling persistente (constante) gegevens, die wordt gebruikt door de applicatiesystemen van een bepaalde onderneming, en die wordt beheerd door een databasebeheersysteem (dbms)
- **Databaseserver** ofwel **Database Management System (DBMS)** is een verzameling programma's waarmee gebruikers een database kunnen maken en onderhouden
- **Relationele database** is een database die bestaat uit tabellen die onderling een relatie met elkaar hebben (**RDBMS = Relational Database Management System**)
- **Databasetalen** zijn speciale talen waarmee opdrachten aan een databaseserver geformuleerd kunnen worden
- **SQL (Structured Query Language)** is een relationele database taal, die instructies bevat voor het invoeren, wijzigen, verwijderen, raadplegen en beveiligen van gegevens
- **Query** ofwel **SQL Statement** is een opdracht (commando) aan een database om gegevens toe te voegen, op te halen, aan te passen of te verwijderen

Het woord **database** wordt in de praktijk voor verschillende begrippen gebruikt.

- Opgeslagen gegevens (database).

- De manier waarop gegevens zijn opgeslagen (datamodel).
- De software om databases te maken en te benaderen (Database Management Systeem of DBMS).

In dit lesmateriaal betekent **database** altijd ‘de opgeslagen gegevens’. Een database is een archief die digitaal is opgeslagen en ingericht om flexibel te raadplegen en te gebruiken.

Databases spelen een belangrijke rol bij het archiveren en actueel houden van gegevens bij onder meer de overheid, financiële instellingen/bedrijven, in de wetenschap en worden daarnaast op kleinere schaal ook privé gebruikt.

Een database is pas een database als het aan de minimale (**CRUD**) voorwaarden voldoet:

- Gegevens moeten kunnen worden opgeslagen. (**Create**)
- Gegevens moeten kunnen worden opgezocht en doorzocht. (**Read**)
- Gegevens moeten gewijzigd kunnen worden. (**Update**)
- Gegevens moeten verwijderd kunnen worden zonder dat dat de werking van het systeem nadelig beïnvloedt. (**Delete**)

Om een bruikbare database te hebben moet aan een aantal voorwaarden worden voldaan:

- Gegevens moeten **consistent** zijn (ze mogen elkaar niet tegenspreken). Als twee verschillende databasegegevens elkaar tegenspreken wordt dit **inconsistentie** van gegevens genoemd.
- Om dit te waarborgen is het de bedoeling om gegevens niet dubbel (**redundant**) op te slaan. De gegevens van de locatie van een klas wordt niet bij elke student in de studententabel ingevoerd, maar staat alleen in de tabel klasgegevens.
- De samenhang of de relatie met andere gegevens moet blijven kloppen. Een student mag geen niet-bestante klas toegewezen krijgen. Een klas waar nog studenten in zitten mag niet uit de database worden verwijderd. Dit heet **referentiële integriteit**.

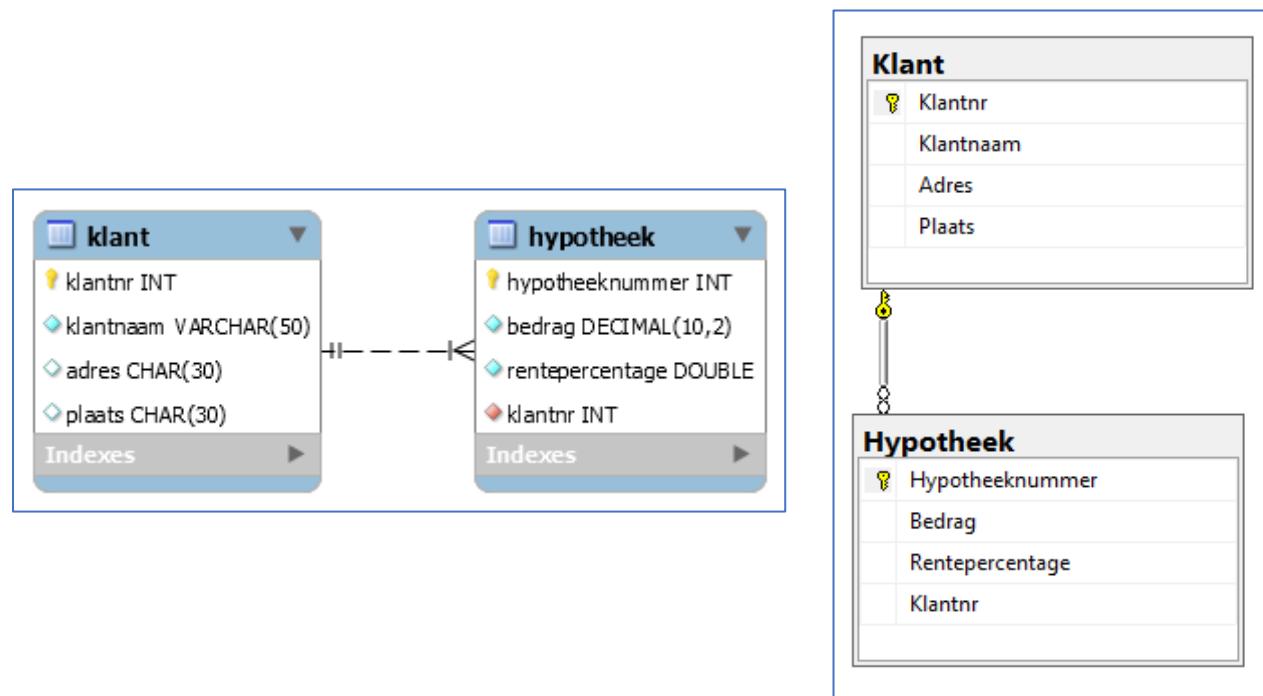
Een database moet **integer** zijn. Integere gegevens zijn tijdig, juist, volledig en geautoriseerd:

- **Tijdig:** als de betaling van een klant pas na een jaar ingevoerd wordt, heeft hij ten onrechte een deurwaarder op bezoek gehad.
- **Juist:** als de naam van een klant verkeerd gespeld wordt, is hij beledigd of is de rekening niet rechtsgeldig.  
De juistheid wordt ook wel geformuleerd als “in overeenstemming met de werkelijkheid”.  
Als bijvoorbeeld een werknemer maar voor één afdeling tegelijk mag werken, mag nergens in de database geregistreerd kunnen worden dat deze werknemer voor twee of meer afdelingen werkt. Hierbij geldt ook het bekende GIGO principe: Garbage In, Garbage Out. Als de invoer van gegevens niet goed is, komt er rommel uit het systeem.
- **Volledig:** als een adres en woonplaats niet ingevuld zijn, kan de postbode de brief niet bezorgen.
- **Geautoriseerd:** als iedereen de salarisgegevens kan aanpassen, kan het gebeuren dat iemand opeens een riante salarisverhoging heeft gekregen. De gegevens moeten daarom geautoriseerd toegevoegd, onderhouden of verwijderd worden.

### 3.3 Datamodel

Met een **datamodel** (of **gegevensmodel** of **databasediagram** of **Entity Relation Diagram (ERD)**) wordt beschreven hoe de gegevens in een informatiesysteem gestructureerd zijn. In een datamodel kun je zien welke gegevens in een informatiesysteem worden vastgelegd en wat de verbanden zijn tussen deze gegevens.

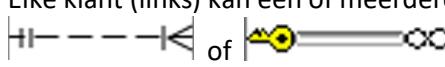
Hieronder staat op twee verschillende manieren een datamodel van een database met twee tabellen weergegeven. Links een diagram zoals gebruikt door Oracle en MySQL en rechts een diagram zoals gebruikt door Microsoft SQL Server.



Een datamodel bestaat uit:

- tabellen, de namen van de tabellen zijn hier **klant** en **hypotheek**
- velden (ook wel kolommen genoemd), in de tabel **klant** zijn dit **klantnr**, **klantnaam**, **adres** en **plaats**
- sleutels, de velden waarbij een sleutel afgebeeld staat, **klantnr** is de primaire sleutel in de tabel **klant**. Dit veld zorgt ervoor dat er geen dubbele records in de tabel ingevoerd kunnen worden, dit veld heeft voor ieder record een andere (unieke) waarde.
- de **relaties** tussen de tabellen, de lijn met het sleutelsymbool en het oneindig symbool ∞ of één en een harkje. Eén klant kan veel hypotheken afsluiten (voor haar huis, haar 2<sup>e</sup> huis, haar boot etc.).

Bovenstaand datamodel (Entity Relation Diagram) kunnen we als volgt lezen:

- Elke klant (links) kan één of meerdere hypotheken (rechts) hebben
  - of
  - 
- Elke hypothek behoort tot slechts één klant, dat betekent:  
Er is één hypothek zonder klant en er is één hypothek van meerdere klanten
- Een hypothek kan pas in tabel hypothek ingevoerd worden nadat de klant in tabel klant ingevoerd is (anders zou er een hypothek ontstaan zonder klant)

- Een klant kan wél ingevoerd worden voordat er een hypotheek voor de klant wordt ingevoerd (een nieuwe klant die nog geen hypotheek heeft)

### 3.4 Tabellen, velden of kolommen, records

Gegevens in een relationele database worden bewaard in tabellen.

Hieronder staat een voorbeeld van de klant tabel. Deze tabel bevat gegevens over drie klanten.

Klantnr	Klantnaam	Adres	Plaats
1	Pjotr	Dijkweg 6	Rotterdam
2	Katinka	Kerkstraat 316	Amsterdam
3	Ronald	Jonkerstraat 80	Den Haag

**Klantnr, Klantnaam, Adres en Plaats** zijn de namen van de kolommen of velden in de tabel. De kolom **Klantnr** bevat de waarden 1 , 2 en 3. De tabel **klant** bevat drie records of rijen, één voor elke klant.

Een tabel heeft een aantal speciale eigenschappen:

- De kruising van een rij en een kolom mag maar één waarde bevatten. Als een klant twee adressen heeft, kan er maar één adres in deze tabel opgeslagen worden.
- De rijen in een tabel hebben geen specifieke volgorde. Je kunt dus nooit spreken over de eerste rij, de laatste drie of de volgende rij. De inhoud van een tabel is een verzameling rijen.
- Applicaties hoeven niet altijd opnieuw geprogrammeerd te worden als er een kolom aan een tabel wordt toegevoegd, bijvoorbeeld als er een kolom factuuradres aan toe gevoegd zou worden heeft dat geen invloed op applicaties die dit gegeven niet nodig hebben.

### 3.5 Primaire sleutel / primary key

De primaire sleutel van een tabel is een kolom in een tabel (of een combinatie van een aantal kolommen) die gebruikt kan worden als unieke identificatie voor de rijen in die tabel.

Twee verschillende rijen in een tabel mogen dus nooit in de primaire sleutel dezelfde waarde hebben en in elke rij moet de primaire sleutel altijd één waarde hebben. De Klantnr-kolom in de eerder getoonde tabel klant is gekozen als de primaire sleutel van de tabel. Twee klanten mogen dus nooit hetzelfde nummer hebben en er zal nooit een klant mogen zijn zonder klannummer.

### 3.6 Refererende sleutel / foreign key

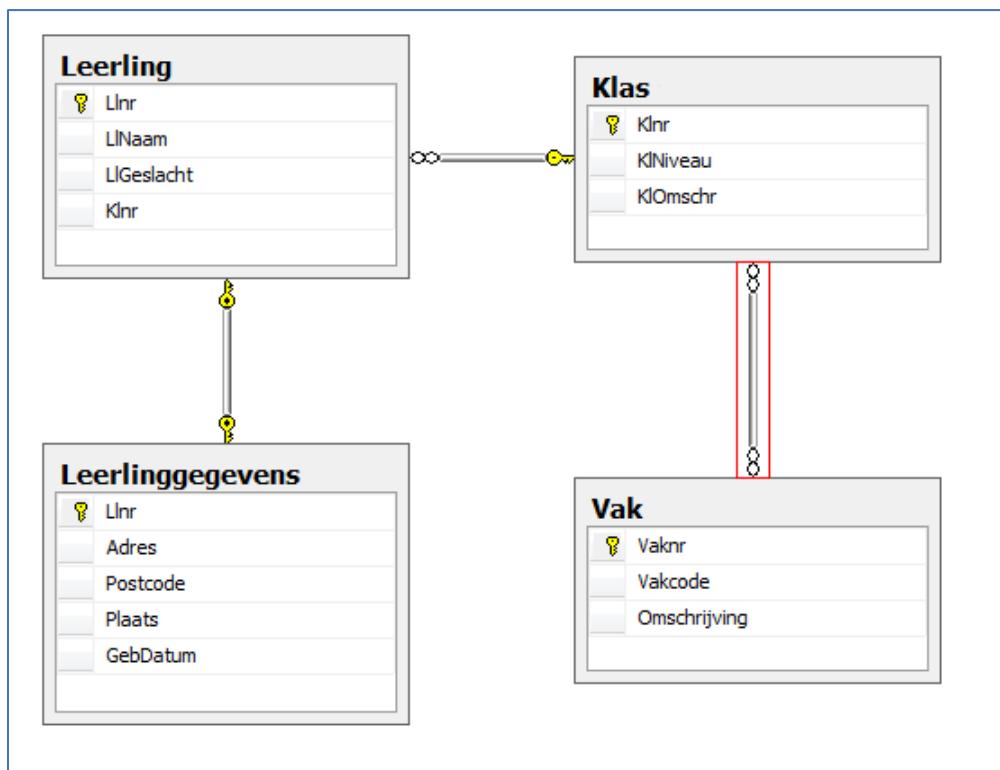
Een refererende sleutel (foreign key; deze term zal in de rest van de reader gebruikt worden) is een kolom (of combinatie van kolommen) in een tabel die verwijst naar de primaire sleutel van een andere tabel. In de tabel hypothetiek is het veld klantnr de foreign key, die verwijst naar de primaire sleutel klantnr in de tabel klant.

### 3.7 Soorten relaties

Binnen een relationele database heeft elke tabel een relatie met één of meer andere tabellen. Er zijn vier verschillende relaties die je kunt tegenkomen.

- 1 op 1
  - 1 op n (= 1 op veel)
  - n op 1 (= veel op 1)
  - n op m (= veel op veel)
- 

Het volgende plaatje toont een stukje van een databasediagram waarin de verschillende relatiesoorten voorkomen.



De relatie die je het meest tegenkomt binnen databases, is de ‘1-op-veel’- of ‘veel-op-1’-relatie. In het bovenstaande databasediagram ligt er tussen de tabellen *Leerling* en *Klas* een dergelijke relatie.

- Een klas kan diverse leerlingen bevatten
- Een leerling kan slechts in één klas zitten

Een ‘1-op-1’-relatie kom je zelden tegen in een database. Het kan wel, maar vaak is dit een teken dat het databaseontwerp nog verbeterd dient te worden. In het voorbeeld ligt deze relatie tussen de tabellen *Leerling* en *Leerling gegevens*.

- Een leerling heeft een adres, postcode, geboortedatum en dergelijke.

De ‘veel-op-veel’-relatie (rood) is wel getekend, maar kan standaard niet worden aangemaakt in SQL.

- Een klas volgt meerdere vakken
- Een vak wordt gevuld door meerdere klassen

Een ‘veel-op-veel’-relatie kan wel uit een ontwerp komen, maar kan in de praktijk NOoit in een database worden gebouwd. Hiervoor moet gebruik worden gemaakt van een zogenaamde **koppeltabel**. In deze koppeltabel zitten de primaire sleutelvelden van beide tabellen die worden gekoppeld. Deze velden worden de primaire sleutel van de koppeltabel.

Hiernaast zie je de tabellen *Klas* en *Vak*.

Een klas kan een vak volgen, maar een vak kan ook door verschillende klassen gevuld worden. Hier is dus sprake van een ‘veel-op-veel’-relatie. Dit mag en kan niet in een database.

Een koppeltabel moet daarom tussen de tabellen *Klas* en *Vak* geplaatst worden.



In de koppeltabel zijn ook nog twee velden toegevoegd, waardoor er een rooster kan worden gegenereerd.

### 3.8 Database Management System (DBMS)

opdracht moeten geven. Dit kan jij zijn, als gebruiker, maar het kan ook een programma zijn. In het tweede geval is het programma de gebruiker.

Met behulp van speciale talen worden opdrachten aan een DBMS gegeven. Dit soort talen wordt **databasetalen** genoemd. Opdrachten worden door gebruikers ingevoerd en door het DBMS verwerkt. Elk DBMS, van welke fabrikant dan ook, bezit een databasetaal. Sommige systemen hebben zelfs meer dan één databasetaal. Tussen al die talen bestaan verschillen. Deze verschillende talen zijn te verdelen in groepen. Een van deze groepen wordt gevormd door databasetalen voor relationele databases.

### 3.9 Stand-alone DBMS

Bij kleine bedrijven is de DBMS met een database vaak geïnstalleerd op een PC, zoals bijvoorbeeld in Microsoft Acces. Er is dan slechts één gebruiker tegelijk actief in de database (**Single User database**).

## Stand-alone DBMS



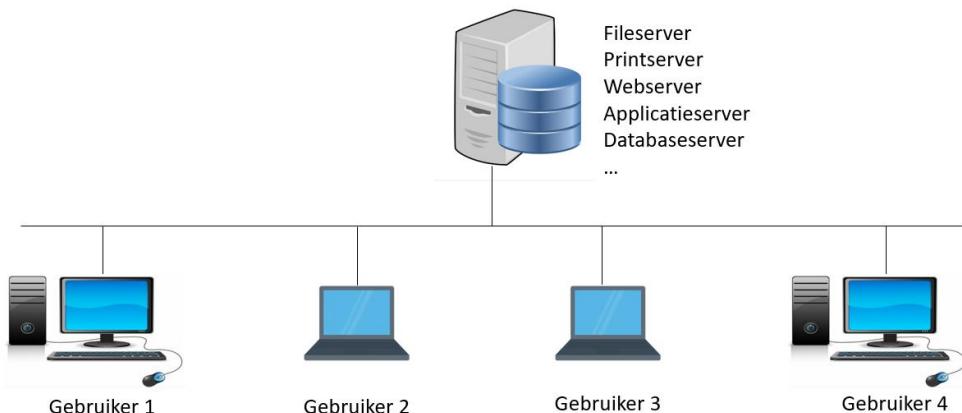
Een **stand-alone database** is relatief eenvoudig te beheren. De werkzaamheden in de database van die ene medewerker komt niet in conflict met werkzaamheden van andere medewerkers, omdat er geen andere medewerkers tegelijkertijd in de database aan het werk zijn.

Stel nu dat de gebruiker bezig is met een aanpassing van gegevens in de database en op dat moment de stroom uitvalt. Ofwel de PC gaat uit en de database is down gebracht. Is deze wijziging dan reeds in de database aangebracht of niet? Er mag in ieder geval nooit een twijfelachtige situatie ontstaan of een situatie waarbij een deel van een actie uitgevoerd is en een ander deel van de actie niet! Microsoft Access zal bij een crash (ineenstorting van het programma) bijvoorbeeld terug gaan naar de laatste consistente situatie die hij kent en zal de incomplete transacties terugdraaien. Dit terugdraaien van transacties wordt een **rollback** genoemd en zullen we in de database beheer lessen verder bestuderen.

### 3.10 DBMS op server

Bij iets grotere bedrijven staat de DBMS met een database vaak geïnstalleerd op een server, zodat meerdere medewerkers via het lokale netwerk gelijktijdig van de database gebruik kunnen maken. Vaak betreft het een server die voor meerdere functionaliteiten wordt gebruikt, zoals fileserver, printserver, webserver, applicatieserver, databaseserver, etc.

## DBMS op server



Zolang alle gebruikers op andere tabellen in de database werken zitten de werkzaamheden van de gebruikers elkaar niet in de weg en kunnen ze allen hun werkzaamheden gelijktijdig in de database uitvoeren. Wanneer echter meerdere gebruikers werkzaamheden willen uitvoeren op eenzelfde tabel dan kunnen de werkzaamheden van de gebruikers elkaar hinderen.

Wanneer meerdere gebruikers gelijktijdig dezelfde gegevens uit een tabel willen lezen is er nog geen probleem en kunnen de gebruikers onafhankelijk van elkaar de gegevens gelijktijdig uit de tabel lezen.

Wanneer echter meerdere gebruikers gelijktijdig gegevens in een tabel willen wijzigen dan ontstaat er mogelijk wel een probleem. Stel dat er in het magazijn een voorraad van een bepaald artikel 80 stuks is en er twee gebruikers gelijktijdig in de database een aanpassing willen maken op de voorraad in de tabel. Zo wil de ene gebruiker bijvoorbeeld de voorraad van een artikel in het magazijn verhogen omdat er net een levering is gedaan van 50 stuks terwijl een andere gebruiker de voorraad

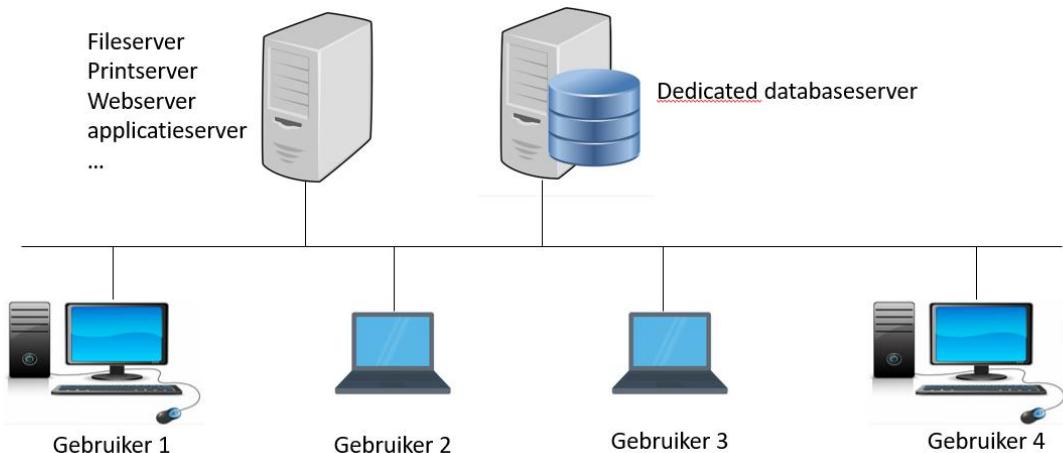
juist wil verlagen omdat er net een aantal van 25 van dat artikel is verkocht. Hoe gaat het DBMS hier dan mee om? Welke actie wordt dan correct uitgevoerd? Er mag in ieder geval nooit een twijfelachtige situatie ontstaan. Het DBMS kent hiervoor een zogenaamd **locking** systeem en zullen we in de database beheer lessen verder bestuderen.

### 3.11 DBMS op dedicated database server

Het nadeel van een DBMS geïnstalleerd op een server die voor meerdere functionaliteiten wordt gebruikt (fileserver, printserver, webserver, applicatieserver, databaseserver, etc.) is dat een zwaar proces voor een andere functionaliteit invloed kan hebben op de prestaties van het DBMS, immers de beschikbare processortijd, het beschikbare geheugen en de toegang tot de schijven van de server zal door het besturingssysteem (**Operation System**) verdeeld moeten worden tussen de verschillende processen.

Om deze afhankelijkheden te voorkomen wordt daarom een DBMS vaak geïnstalleerd op een specifieke databaseserver die géén andere functionaliteiten heeft. Zo'n specifieke database server wordt een **dedicated database server** (toegewijde databaseserver) genoemd.

## DBMS op dedicated database server

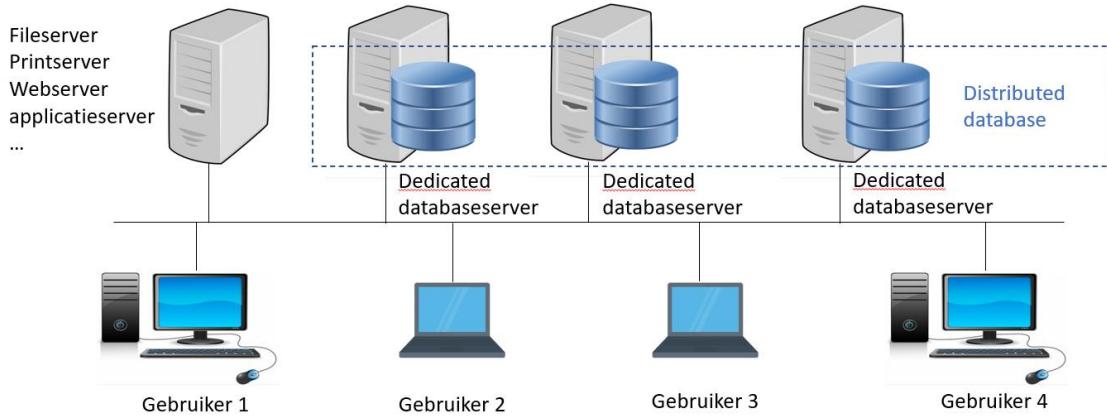


Doordat de DBMS geïnstalleerd is op een dedicated database server kan deze optimaal functioneren onafhankelijk van invloed van andere processen.

### 3.12 Distributed databases

Het kan zijn dat een database zodanig groot wordt dat de server niet meer in staat is om de database te bedienen. In dat geval kan een database in meerdere stukken worden verdeeld, die elk op een aparte databaseserver wordt geïnstalleerd. De aparte databases vormen samen één grote database en gedragen zich naar buiten toe alsof het één database is. Zo'n database bestaande uit meerdere fysieke databases wordt een **Distributed database** genoemd.

## Distributed database

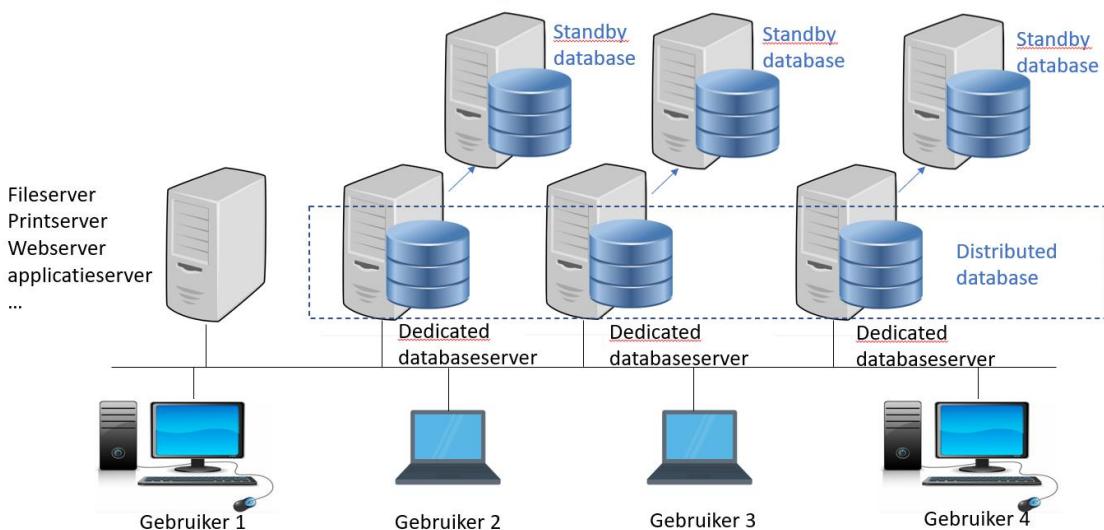


Doordat de gedistribueerde database bestaat uit meerdere databaseservers wordt de database bediend met een processorkracht van alle processoren van alle servers tezamen. Indien het een internationaal gebuikte database is kan er voor gekozen worden om elke server in een bepaald land of wereldeel te plaatsen dicht bij de lokale gebruikers. Omdat de databases onderling met elkaar moeten kunnen communiceren om de data te synchroniseren moet tussen de databaseservers een betrouwbaar netwerk zijn, die onafhankelijk is van overige netwerkactiviteiten.

### 3.13 Standby databases

Indien een databaseserver door een storing down gaat en gerepareerd of vervangen moet worden, dan kan dat vele uren of zelfs dagen tijd kosten, vooral als er dan ook nog een backup van de database moet worden teruggehaald. Dit is voor sommige systemen onacceptabel. Als bijvoorbeeld een database van de luchtverkeersleiding op Schiphol down gaat en het uren of dagen zal kosten voordat de database weer beschikbaar is, dan zouden alle vliegtuigen die nog in de lucht zijn moeten wachten met landen totdat de database weer beschikbaar is. Dat zou dan een onacceptabele situatie zijn.

## Standby database



Als oplossing hiervoor kunnen standby databases worden gebruikt die de productiedatabases op elke transactie volgen. Elke wijziging in de productiedatabase wordt direct doorgegeven en uitgevoerd op de standby database. In geval van een crash van de productiedatabase wordt dan direct de productiedatabase overgenomen door de standby database en is de standby database de productiedatabase geworden.

### 3.14 Complex database beheer

Zoals uit bovenstaande voorbeelden is aangegeven kan het beheren van databases bij grote organisaties erg complex worden. Wat gebeurt er bijvoorbeeld als één van de databases in het plaatje met de distributed database met standby databases crashed en daarna weer wordt opgestart? Ontstaat er dan inconsistentie tussen de databases?

Daar komt nog bij dat de databases verbonden kunnen zijn met externe databases (van andere organisaties) om gegevens tussen de databases te kunnen uitwisselen. Daardoor wordt het geheel van het database landschap nog complexer om te beheren.

Iedere organisatie heeft zijn eigen complexiteit in zijn database landschap.

Waar het nu om gaat is dat wij ons realiseren dat de databasebeheerder voor dat soort situaties scenario's klaar moet hebben en daarin geoefend moet zijn, net zoals een EHBO'er geoefend moet zijn in het reanimeren van mensen.

In de lessen over het beheren van databases in deze reader wordt een inleiding gegeven van de meest belangrijke taken en verantwoordelijkheden van de databasebeheerder.

Voor het beheren van de databases is kennis van de databasetaal SQL noodzakelijk. Daarnaast dient een databasebeheerder een gebruikersvraag voor een rapport of een wijziging te kunnen vertalen naar een technische vraag die met behulp van SQL kan worden opgelost en m.b.v. SQL rapporten genereren.

We beginnen daarom in de komende hoofdstukken met het leren van de databasetaal SQL.

## Hoofdstuk 4 SQL

### 4.1 ANSI SQL

SQL is een ANSI/ISO-standaardtaal voor een relationeel databasemanagementsysteem (RDBMS). Het is een gestandaardiseerde taal die gebruikt kan worden voor taken zoals het bevragen en het aanpassen van gegevens in een relationele database. SQL kan met vrijwel alle moderne relationele databaseproducten worden gebruikt.

Naast de standaard-SQL breiden DBMS-fabrikanten de taal nog uit met eigen specifieke opdrachten. Deze opdrachten zijn meestal niet uitwisselbaar met andere database omgevingen.

Sommige systemen zijn hoofdlettergevoelig (vaak gebaseerd op Unix of Linux), andere zijn dit niet (Microsoft).

### 4.2 Geschiedenis van SQL

De geschiedenis van SQL begint als de Britse computerwetenschapper Dr. Edgar F. Codd in juni 1970 een artikel publiceert met als naam 'A Relational Model of Data for Large Shared Data Banks'. De inhoud van dit artikel wordt vervolgens geaccepteerd als het uiteindelijke model voor relationele databases.

Aan de hand van dit relationele model bouwt IBM een experimenteel systeem, genaamd 'System R'.

Donald Chamberlin en Raymond Boyce ontwikkelen de taal 'Structured English Query Language', of SEQUEL, omdat er ook een mogelijkheid moet zijn om data in 'System R' te lezen en te bewerken. Helaas zat er aan het woord SEQUEL een geregistreerd handelsmerk van een Engelse vliegtuigbouwer en daarom is, om de concepten van de taal te kunnen publiceren, de naam veranderd naar SQL.

Hoewel er na publicatie van het artikel van Codd een aantal relationele databases verschijnt, maakt IBM zich in 1978 op om het eerste relationele databasesysteem met SQL als onderdeel van System/38 op de markt te zetten. Als dit in 1979 daadwerkelijk gebeurt, blijkt IBM niet meer de eerste te zijn. Relational Software is IBM een paar weken te vlug af met de introductie van Oracle. Enkele jaren later neemt Relational Software de naam Oracle over als bedrijfsnaam.

### 4.3 Soorten SQL-opdrachten

De databasetaal SQL is op twee manieren te gebruiken:

- **Interactief**  
SQL instructies worden door gebruiker ter plekke ingevoerd en direct door de database server verwerkt
- **Voorprogrammeerd SQL**  
SQL instructies zijn opgenomen in een programma dat in een andere programmeertaal geschreven is (embedded SQL, Call Level Interface SQL)

In de komende hoofdstukken zullen wij ons bezig houden met het interactief gebruik van SQL.

## 4.4 Drie soorten SQL opdrachten

We onderscheiden drie soorten SQL opdrachten:

### 1. Data Definition Language (DDL)

Gebruikt door databasebeheerder (Database Administrator DBA) voor het aanmaken, wijzigen of verwijderen van objecten (tabellen, views, indexen, etc) in de database.  
B.v. het maken van de tabel Studenten: **CREATE TABLE Studenten...**;

### 2. Data Control Language (DCL)

Gebruikt door databasebeheerder (Database Administrator DBA) om rechten toe te kennen aan gebruikers zodat elke gebruiker toegang heeft tot de objecten waar hij/zij voor zijn/haar functie bevoegd is.

B.v. het toekennen van leesrechten: **GRANT SELECT ON Studenten TO Gebruiker1**;

### 3. Data Manipulation Language (DML)

(Via applicatie) gebruikt door gebruikers voor het invoeren, wijzigen of verwijderen van gegevens in de tabellen in de database.

B.v. het selecteren van gegevens uit een tabel: **SELECT \* FROM Studenten**;

In de komende hoofdstukken zullen wij ons voornamelijk bezig houden met DML opdrachten. We zullen ons vooral bezig houden met het selecteren van gegevens uit tabellen m.b.v. de DML opdracht **SELECT**.

Af en toe zullen we gebruik maken van DML opdrachten om tabellen aan te maken, aan te passen of te verwijderen

Bij het beheren van databases zullen we gebruik maken van DCL opdrachten om gebruikers de nodige privileges op tabellen te geven.

## 4.5 T-SQL

**T\_SQL (Transact-SQL)** is het Microsoft SQL Server dialect van de ISO en ANSI standaarden voor SQL.

In deze reader maken we zo veel als mogelijk gebruik van de ANSI SQL STANDARD, echter daar we werken met Microsoft SQL Server zijn we op enkele plaatsen genoodzaakt om ons te richten op T-SQL. Dat betekent dat bepaalde voorbeelden in deze reader niet zullen gaan werken op andere RDBMS systemen zoals bij voorbeeld Oracle, DB2 of MySQL.

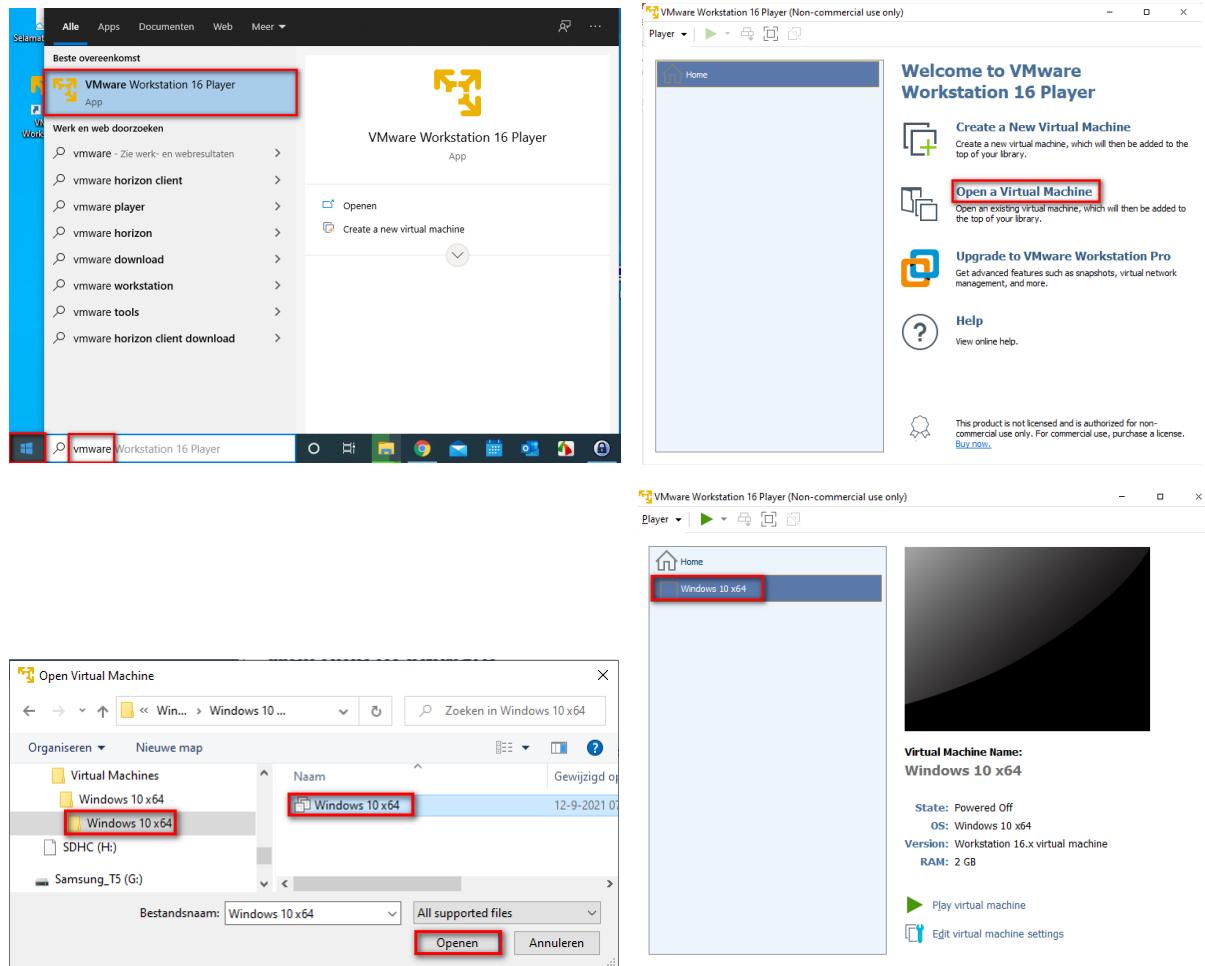
## Hoofdstuk 5 SSMS Graphical User Interface

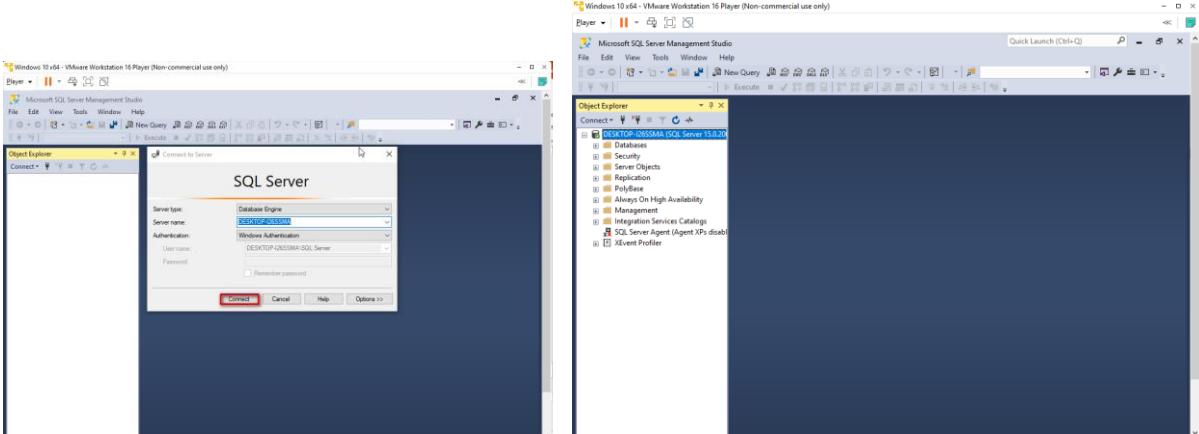
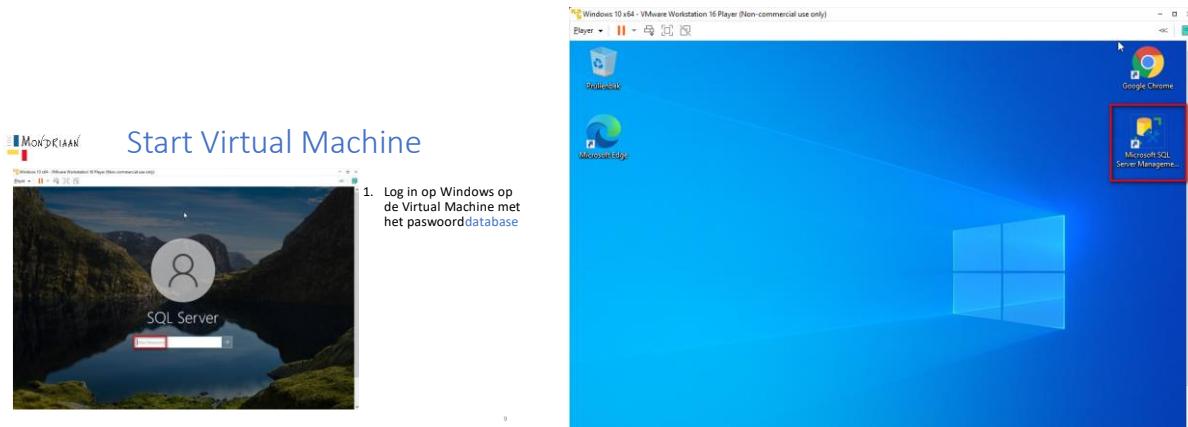
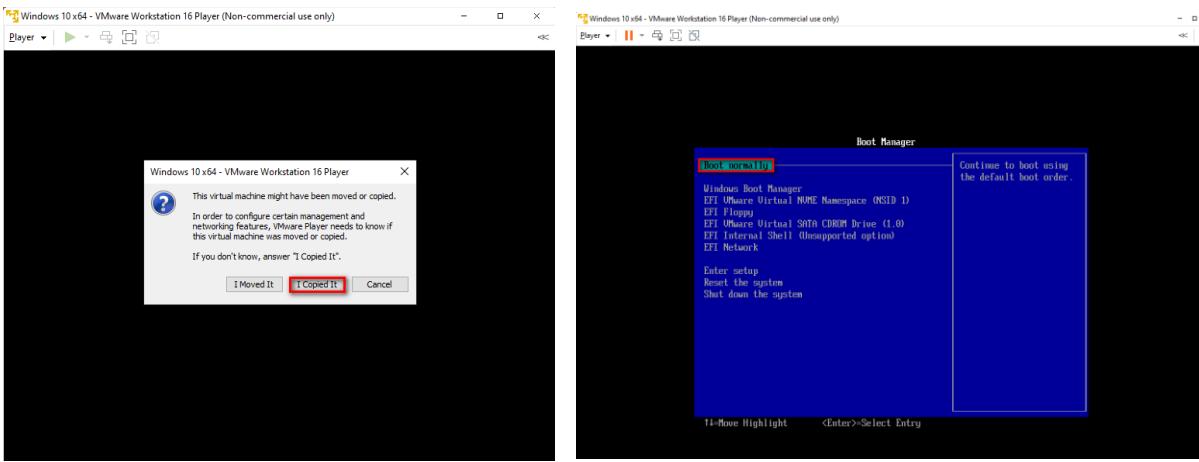
### 5.1 Opstarten van SSMS

**SQL Server Management Studio (SSMS)** is een tool van SQL Server om verbinding te maken met de Microsoft SQL Server DBMS.

In dit hoofdstuk wordt kennis gemaakt met de Graphical User Interface van SSMS en wordt uitgegaan van SQL Server die geïnstalleerd is in de bij deze hoofdstukken gebruikte Virtual Machine.

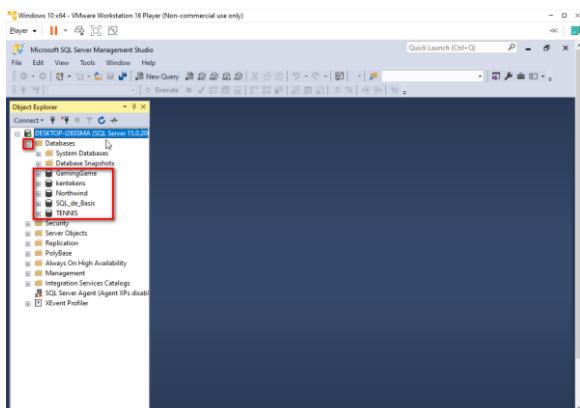
Open de Virtual Machine en start SSMS met Start -> type VMWare -> VMWare Workstation -> Open a Virtual Machine -> zoek de Virtual Machine -> dubbelklik op de Virtual Machine -> I copied it -> klik op VMWare window en druk op Enter -> password = database -> Microsoft SQL Server Management Studio 18 -> Connect





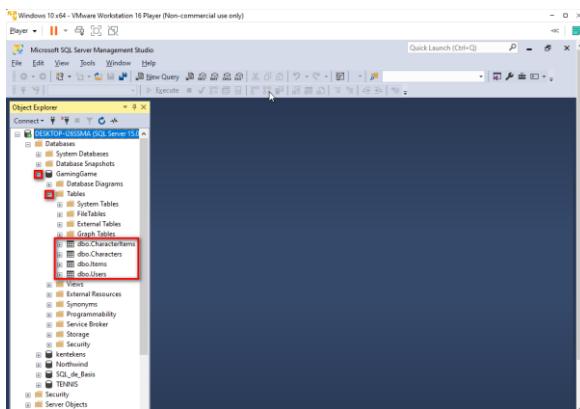
## 5.2 Databases zichtbaar maken in SSMS

De in de DBMS aanwezige databases kan worden getoond door op het kruisje naast Databases te klikken:



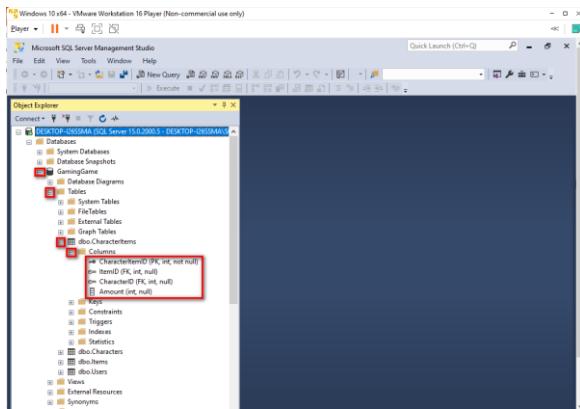
## 5.3 Tabellen zichtbaar maken in SSMS

De aanwezige tabellen in de database kan worden getoond door op het kruisje naast tables te klikken



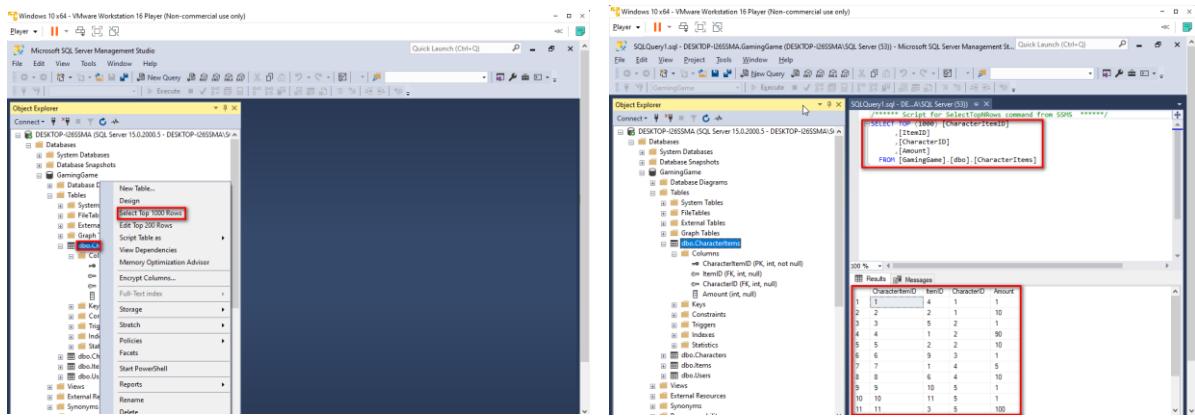
## 5.4 Kolommen zichtbaar maken in SSMS

De kolommen van een tabel kan worden getoond door op het kruisje naast Columns te klikken:



## 5.5 Gegevens in tabel zichtbaar maken in SSMS

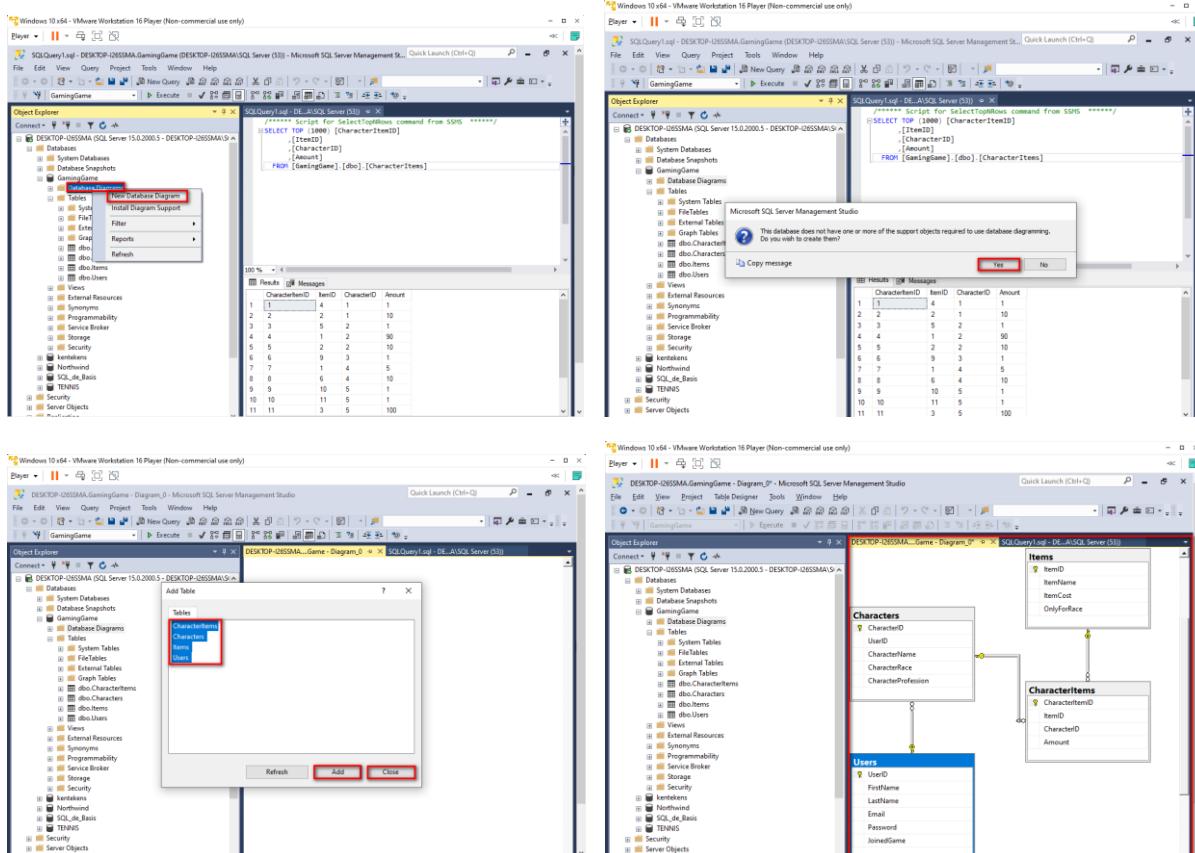
De (eerste 1000 rijen) gegevens in de tabel kan worden getoond door rechtermuisklik op de tabel -> Select Top 1000 rows:

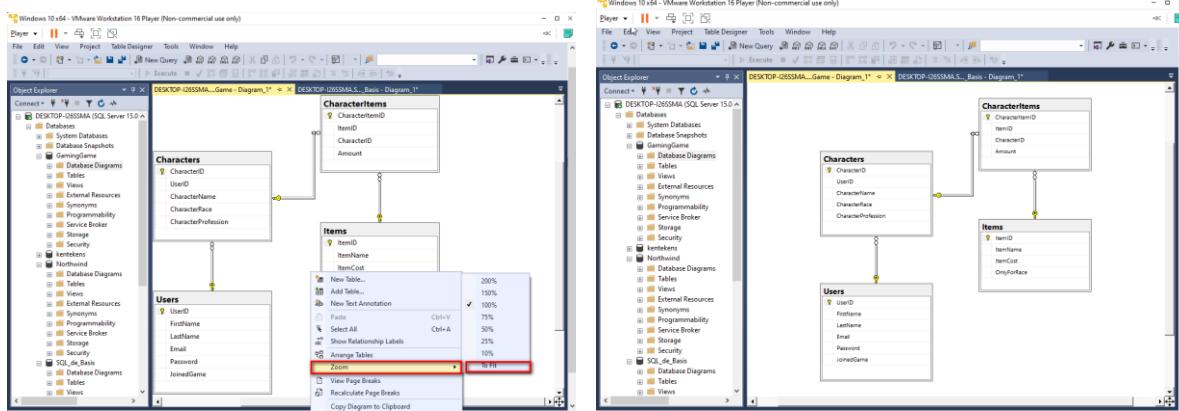


Rechtsboven komt het SQL statement (query) te staan dat het heeft uitgevoerd en rechtsonder komt het resultaat van de query.

## 5.6 ERD zichtbaar maken in SSMS

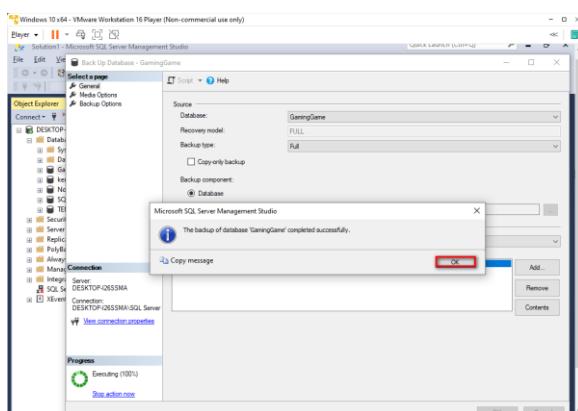
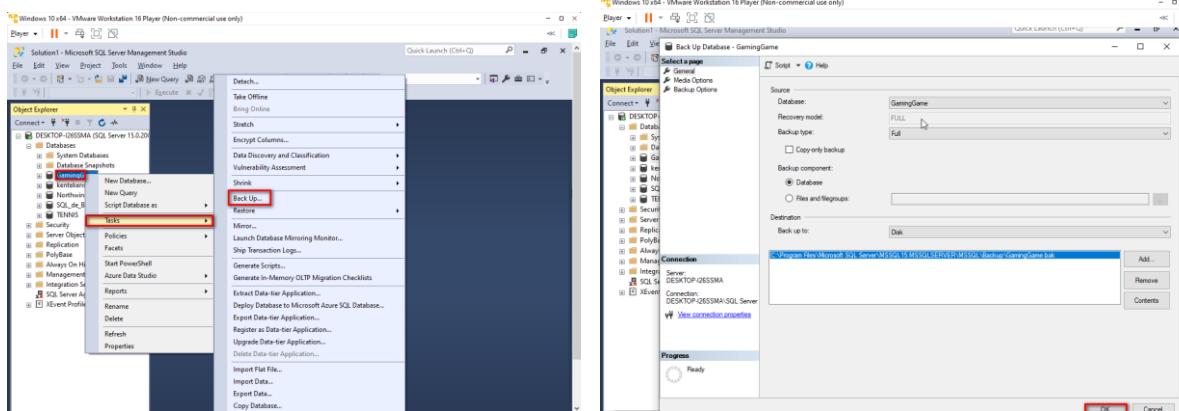
Het Entity Relation Diagram (ERD) van de database kan worden getoond door rechtermuisklik op de database -> New Database Diagram > Yes -> selecteer alle tabellen -> Add -> Close -> rechtermuisklik op tekening -> Zoom -> To Fit





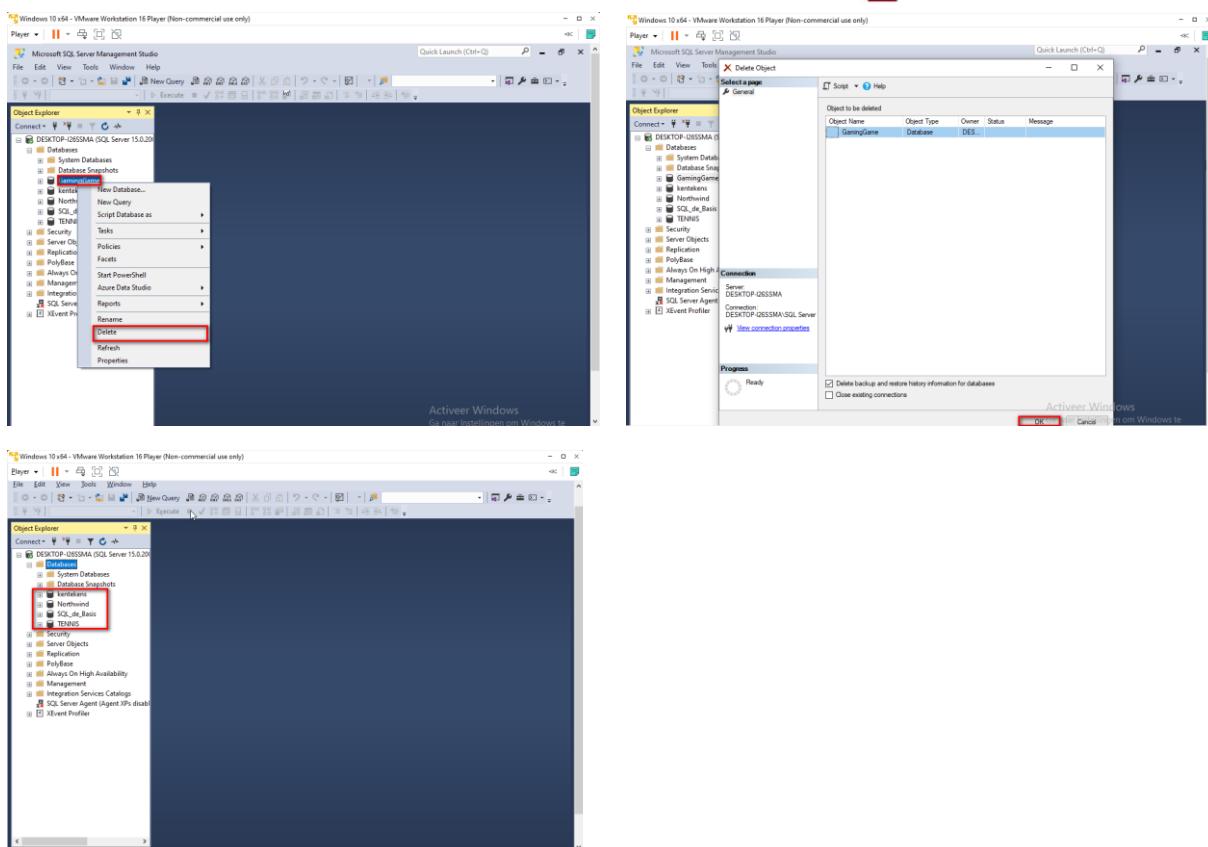
## 5.7 Database backup maken in SSMS

We kunnen een backup van een database maken door rechtermuisklik op database -> Tasks -> Back Up... -> OK



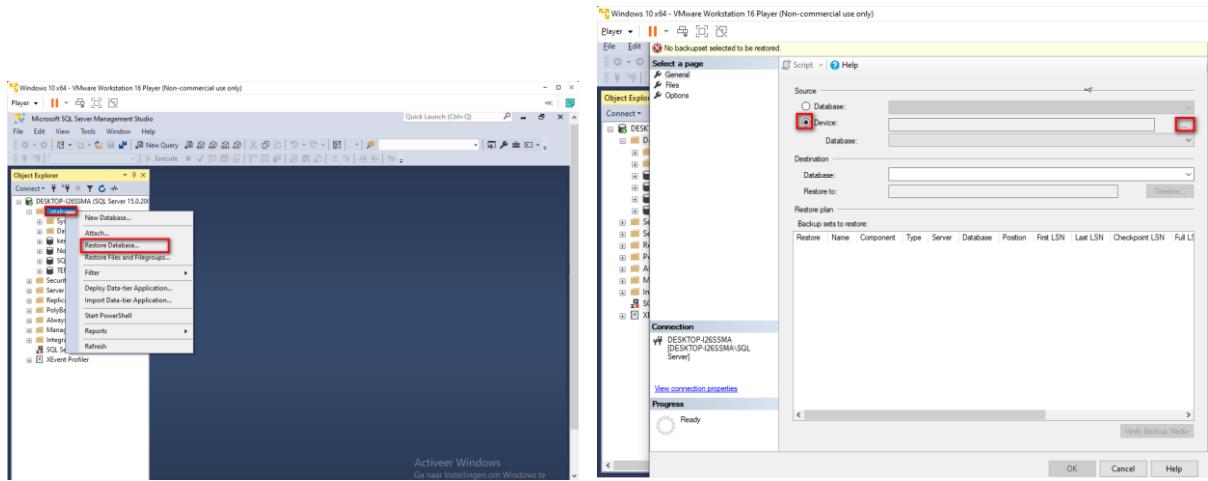
## 5.8 Database verwijderen in SSMS

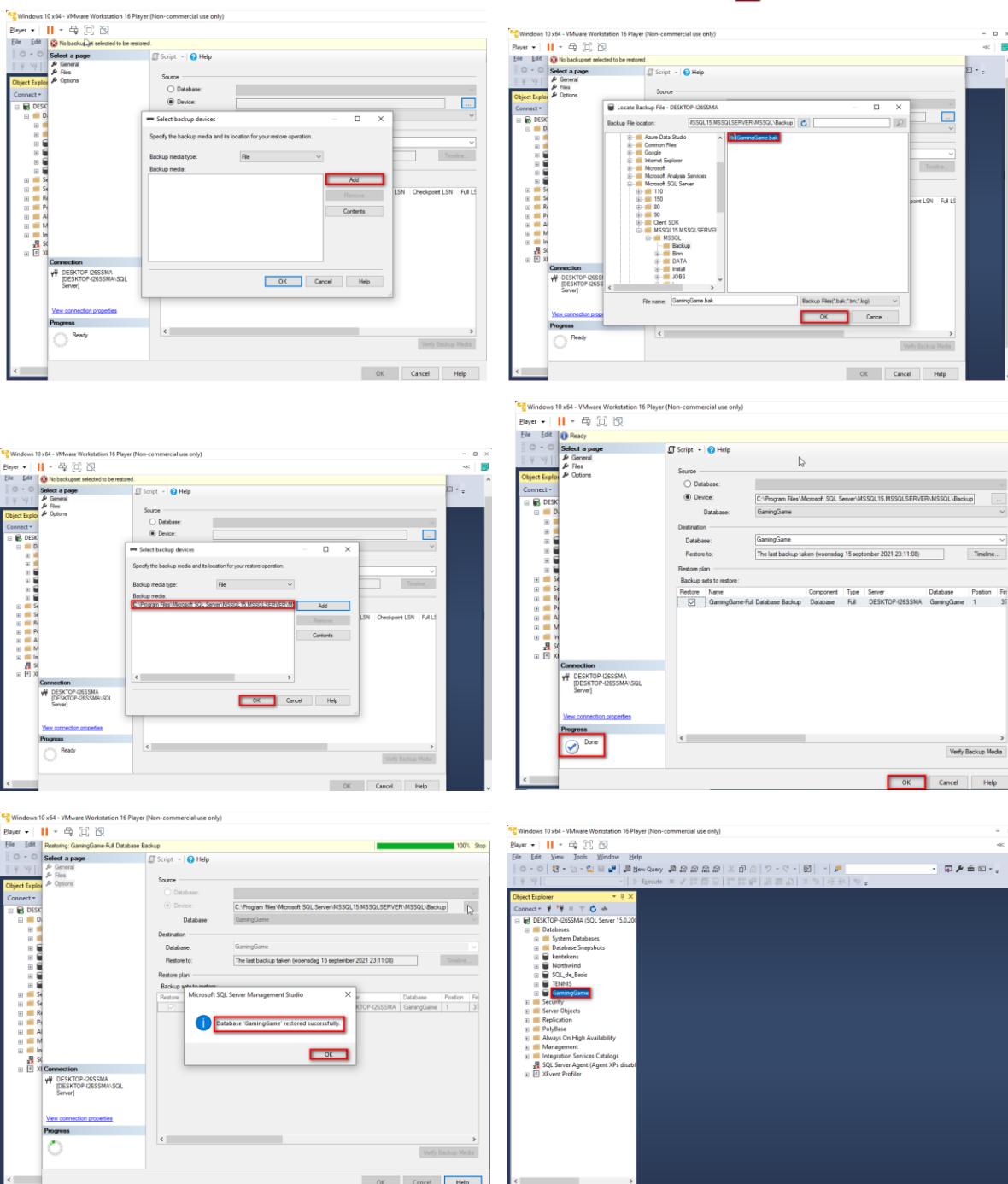
We kunnen een database verwijderen door rechtermuisklik op database -> Delete -> OK



## 5.9 Database uit backup terughalen in SSMS

We kunnen een database uit een backup terughalen door rechtermuisklik op Databases -> Restore Database -> Device -> Add -> selecteer de database backup file (.bak) -> controleer het pad -> OK -> OK -> OK





## 5.10 Courante database

Een database met tabellen is te vergelijken met een kast met ordners. In de tabellen (ordners) zitten de gegevens. Voordat je de gegevens in een tabel (ordner) kunt bekijken of kunt wijzigen moet je eerst de betreffende database courant maken (kast openen). De geopende database (kast) heet de **courante database**. Het is niet mogelijk om meerdere databases tegelijk courant te houden, er kan slechts één database courant zijn. Om gegevens uit een andere database te bekijken of te wijzigen moet eerst de betreffende database courant worden gemaakt. In SQL Server kan een database courant worden gemaakt d.m.v. het **USE <databasenaam>**; commando.

## Slechts één Courante database



### 5.11 Query blad in SSMS

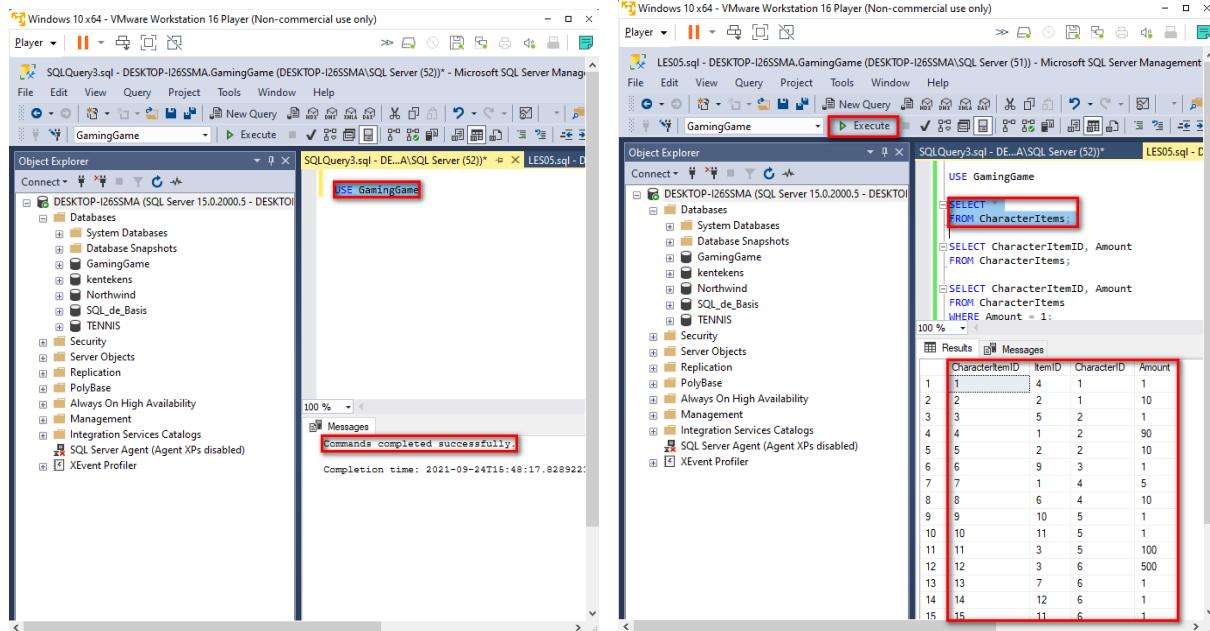
Om SQL commando's in SSMS te kunnen gebruiken dienen we een SQL Query blad te openen door te klikken op de **New Query** knop. Er wordt dan aan de rechterzijde een nieuw SQL Query blad geopend, waarin de SQL commando's kunnen worden ingetypt. Door een SQL statement in het Query blad te selecteren en op de knop **Execute** te drukken wordt het geselecteerde SQL statement uitgevoerd.

CharacterItemID	Amount
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13

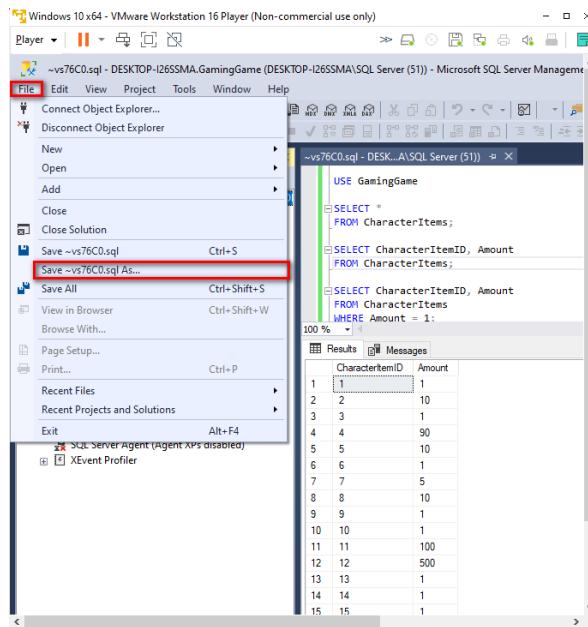
Let op: Indien er géén SQL statement geselecteerd wordt voordat er op de Execute knop wordt gedrukt, dan zullen alle SQL statements op het gehele SQL query blad uitgevoerd worden!

Met het USE commando kan een database courant worden gemaakt, bv: **USE GamingGame;** Rechts onderin komt dan als resultaat een melding dat het commando succesvol is uitgevoerd. Nadat de database courant is gemaakt kan met een SELECT statement gegevens uit een tabel worden

geselecteerd, b.v. **SELECT \* FROM CharacterItems;** Elk SQL statement wordt afgesloten met een ; teken. Druk na het selecteren van de query/ queries die je wilt uitvoeren op de knop Execute. Het resultaat van de query **SELECT \*** bevat door het \* teken alle kolommen van de tabel.



De SQL statements in het SQL Query blad kunnen we in een .sql bestand opslaan door te klikken op File -> Save ... As ... -> selecteer map -> voer bestandsnaam in -> Save



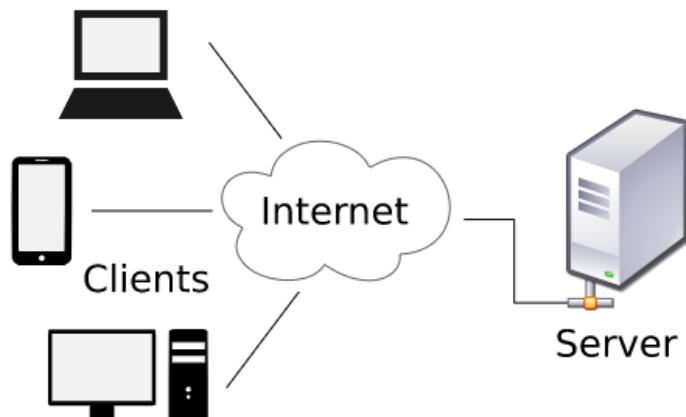
## 5.12 Client-Server Model (NEW)

De meeste database systemen hebben een Client-Servermodel. Het Client-Servermodel is een gedistribueerde applicatiestructuur waarbij taken en workload (werkdruk) verdeeld is tussen de

providers (leveranciers) van een resource of service (**Servers** genoemd) en de serviceaanvragers (**Clients** genoemd).

Vaak communiceren clients en servers via een computernetwerk op afzonderlijke hardware, maar zowel client als server kunnen zich ook in hetzelfde systeem bevinden.

Een serverhost voert één of meer serverprogramma's uit, die hun bronnen delen met clients. Een client deelt meestal geen van zijn bronnen, maar vraagt om inhoud of service van een server. Clients starten daarom communicatiesessies met servers, die wachten op inkomende verzoeken.



Het kenmerk "client-server" beschrijft de relatie van samenwerkende programma's in een applicatie. De servercomponent biedt een functie of service aan één of meerdere clients, die verzoeken om dergelijke services initiëren.

Of een computer een client, een server of beide is, wordt bepaald door de aard van de toepassing die de servicefuncties vereist. De clientsoftware kan ook communiceren met serversoftware binnen dezelfde computer.

Clients en servers wisselen berichten uit in een request-response-berichtenpatroon. De client stuurt een verzoek en de server retourneert een antwoord. Deze uitwisseling van berichten is een voorbeeld van communicatie tussen processen. Om te kunnen communiceren, moeten de computers een gemeenschappelijke taal hebben en moeten ze regels volgen, zodat zowel de client als de server weten wat ze kunnen verwachten. De taal en communicatieregels zijn vastgelegd in een communicatieprotocol.

Een server kan in korte tijd verzoeken van veel verschillende clients ontvangen. Een computer kan op elk moment slechts een beperkt aantal taken uitvoeren en is afhankelijk van een planningssysteem om prioriteit te geven aan binnenvloeiende verzoeken van klanten om hieraan tegemoet te komen.

Microsoft SQL Server RDBMS is de Server en SSMS is de Client in het Client-Servermodel. SSMS dient een verzoek in bij het RDBMS. Het RDBMS voert het verzoek uit en stuurt het resultaat naar de Client. De Client zet het resultaat uit de RDBMS om in een gewenst formaat.

## Hoofdstuk 6 Northwind datamodel (NEW)

### 6.1 Northwind datamodel

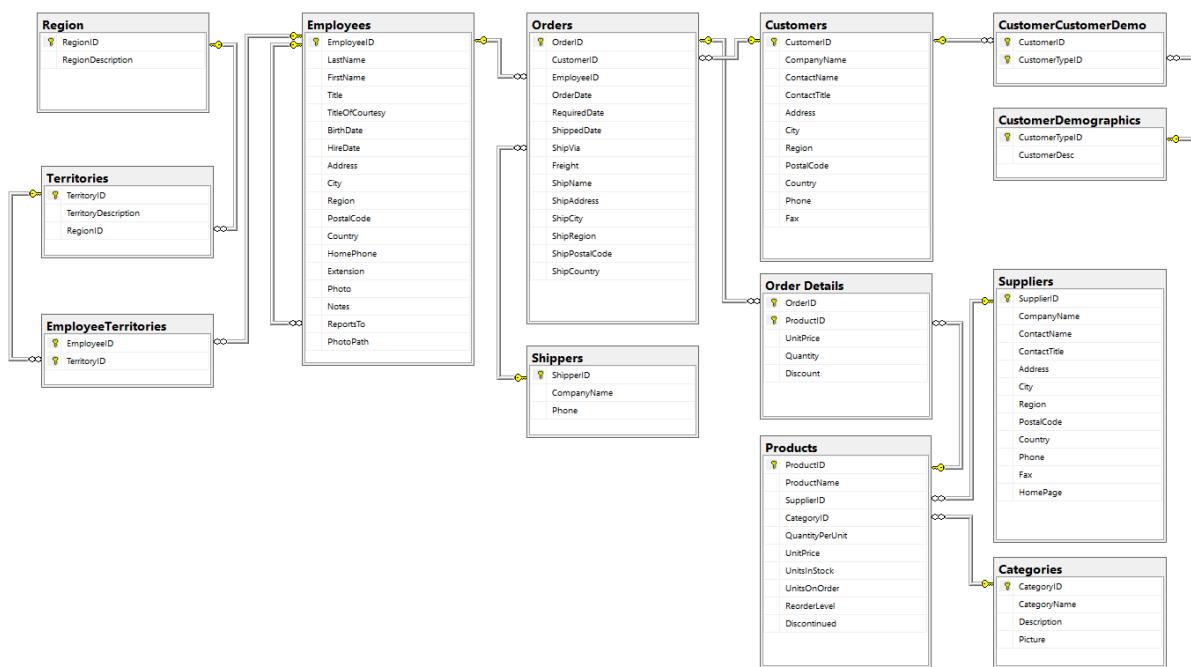
In de voorbeelden en de opgaven van de lessen voor de databasetaal SQL zal in deze reader veel gebruik worden gemaakt van de Northwind database.

De Northwind-database is een voorbeelddatabase die door Microsoft wordt gebruikt om de functies van sommige van zijn producten, waaronder SQL Server en Microsoft Access, te demonstreren.

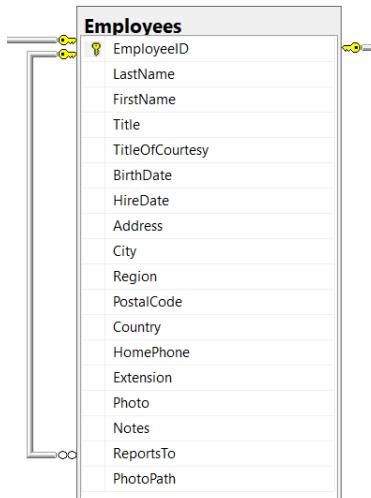
De database bevat de verkoopgegevens voor Northwind Traders, een fictief export/importbedrijf van speciaal voedsel.

Om de uitleg en de oefeningen in de lessen te kunnen begrijpen is het noodzakelijk dat we het datamodel globaal kennen en weten waar we zo nodig gegevens in het datamodel kunnen opzoeken.

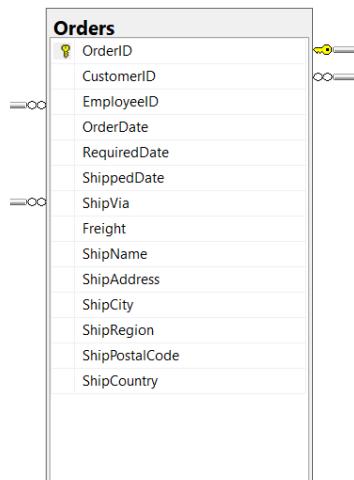
Het datamodel van de Northwind database staat hieronder en in bijlage 2 aangegeven.



Het datamodel toont de tabelstructuur van de Northwind-database. De vierkante blokken (Entiteiten) zijn de tabellen met bovenin de naam van de tabel. In het blok onder de naam van elke tabel staan de namen van de kolommen van de betreffende tabel vermeld. Met de lijnen tussen de tabellen worden de relaties tussen de tabellen weergegeven, waarmee we dus kunnen zien welke tabellen afhankelijk zijn van elkaar. In een latere les zullen we de betekenis van de sleutels , oneindig tekens  en de lijnen  nader bestuderen.

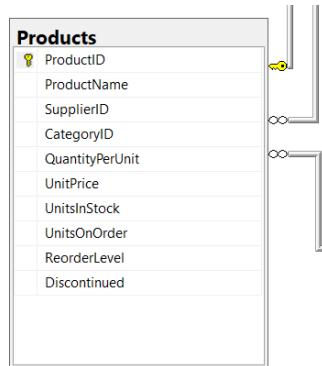


De tabel **Employees** is de tabel met de medewerkers van Northwind Traders. De tabel bevat de persoonlijke gegevens van elke medewerker (personeelsnummer, naam, functie, geboortedatum, datum indiensttreding, adres, personeelsnummer van de manager, etc.).

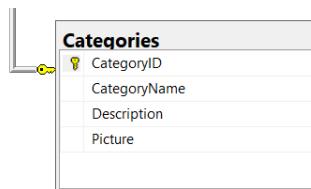


De tabel **Orders** is de tabel met de bestellingen die klanten hebben gedaan. De tabel bevat de gegevens van elke bestelling (bestellingsnummer, klantnummer, personeelsnummer van de verkoper, besteldatum, verzenddatum, verlader, gegevens over het schip, etc.).

De verlader is een organisatie in een logistieke keten die goederen vervoerd wil hebben en daarvoor vervoersbedrijven inschakelt. Elke bestelling wordt dus door een verlader op een schip verzonden.

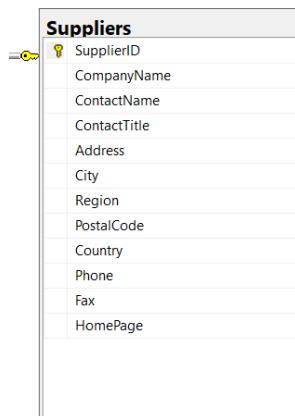


De tabel **Products** is de tabel met alle producten die Northwind Traders verkoopt. De tabel bevat de gegevens van elk product (productnummer, productnaam, leverancier, aantal per unit, prijs per unit, voorraad in het magazijn, aantal in bestelling, aantal te bestellen, etc.).

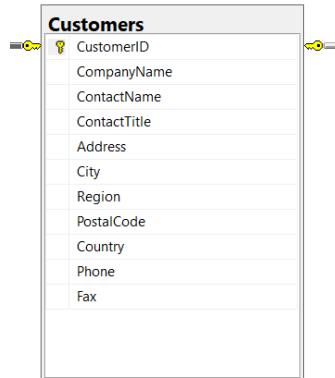


De tabel **Categories** is de tabel met alle categorieën met producten van Northwind Traders (Categorienummer, categorienaam, omschrijving, etc.).

Elk product dat Northwind Traders verkoopt valt binnen één categorie.



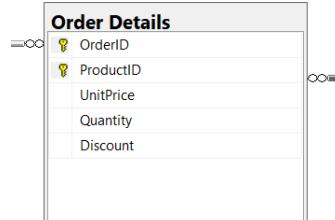
De tabel **Suppliers** is de tabel met alle leveranciers die de producten aan Northwind Traders levert (leveranciernummer, bedrijfsnaam, contactpersoon, adres, etc.).



De tabel **Customers** is de tabel met alle klanten die bestellingen hebben gedaan bij Northwind Traders (klantnummer, bedrijfsnaam, contactnaam, adres, regio, etc.).



De tabel **Shippers** is de tabel met alle verladers die het transport van de bestellingen van Northwind Traders laat uitvoeren door een vervoerder of expediteur (verladersnummer, bedrijfsnaam, etc.).



De tabel **Order Details** is de tabel met alle details van de producten per bestelling (ordernummer, productnummer, prijs per unit, aantal en korting).



De tabel **EmployeeTerritories** is de tabel met de gebieden waarin elke medewerker (als verkoper) actief is (EmployeeID, TerritoryID).



De tabel **Territories** is de tabel met de gebieden waarin medewerkers (als verkopers) actief kunnen zijn (TerritoryID, TerritoryDescription, RegionID).

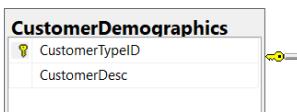


De tabel **Region** is de tabel met de regio's waarin de gebieden (Territories) zich bevinden (RegionID, RegionDescription).



De tabel **CustomerCustomerDemo** is de tabel met de demografische type van elke klant (CustomerID, CustomerTypeID).

Demografische typen zijn beschrijving van de klanten naar bepaalde kenmerken. Deze tabel bevat in de Northwind database géén gegevens.



De tabel **CustomerDemographics** is de tabel met de mogelijke demografische typen en de beschrijving ervan (CustomerTypeID, CustomerDesc). Deze tabel bevat in de Northwind database géén gegevens.

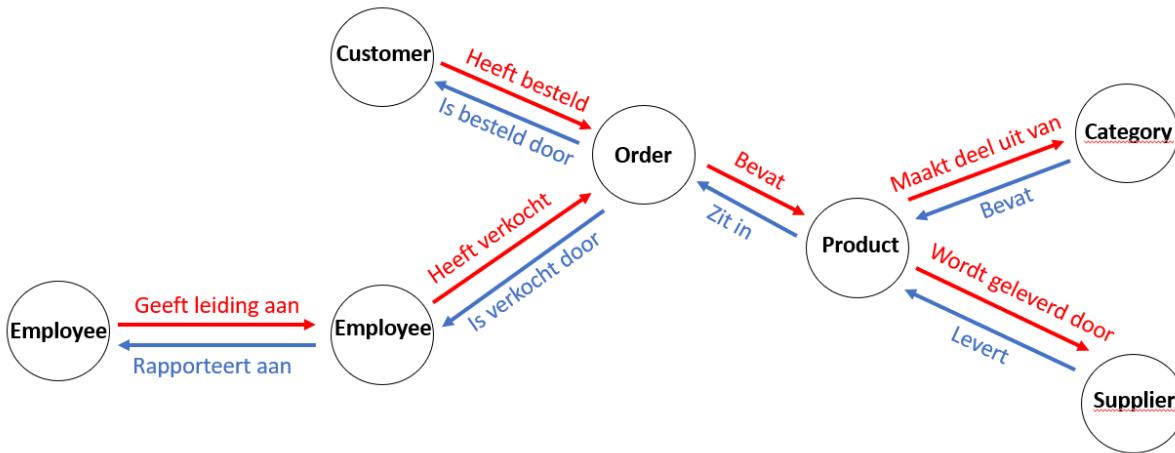
## 6.2 Relaties in het Northwind datamodel (NIAM)

Soms kunnen we alle nodige gegevens uit één tabel halen, maar meestal moeten we gegevens uit meerdere tabellen samenvoegen tot een rapport met de gewenste informatie.

De oorzaak dat we de gegevens uit meerdere tabellen moeten halen is doordat we bij het normaliseren van de tabellen alle gegevensgroepen in aparte tabellen hebben opgesplitst (zie paragraaf 1.2).

In het Northwind datamodel zien we een aantal kerntabellen en een aantal tabellen die de gegevens verder compleet maken. Als we nu alleen naar de kerntabellen kijken dan kunnen we dat in een relatiediagram tekenen waarbij we elke relatie kunnen formuleren in een zin van natuurlijke spreektaal (Natural language Information Analysis Method NIAM):

# Northwind tabelrelaties in NIAM



We kunnen m.b.v. NIAM zeggen over de relatie binnen de tabel Employees:

	EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	BirthDate	HireDate	Address	City	Region	PostalCode	Country	HomePhone	Extension	Photo	Notes	ReportsTo	PhotoPath
1	1	Davolio	Nancy	Sales Representative	Ms.	1948-12-08 00:00:00.000	1992-05-01 00:00:00.000	507 - 20th Ave. E. Apt. 2A	Seattle	WA	98122	USA	(206) 555-9857	5467	0x151C2F00020000000000E014002100FFFFFFFFFF	2	http://accweb/1	
2	2	Fuller	Andrew	Vice President, Sales	Dr.	1952-02-19 00:00:00.000	1992-08-14 00:00:00.000	908 W. Capital Way	Tacoma	WA	98401	USA	(206) 555-9482	3457	0x151C2F00020000000000E014002100FFFFFFFFFF	2	http://accweb/2	
3	3	Leverling	Janet	Sales Representative	Ms.	1963-08-30 00:00:00.000	1992-04-01 00:00:00.000	722 Moss Bay Blvd.	Kirkland	WA	98033	USA	(206) 555-3412	3355	0x151C2F00020000000000E014002100FFFFFFFFFF	2	http://accweb/3	
4	4	Peacock	Margaret	Sales Representative	Mrs.	1937-09-19 00:00:00.000	1993-05-03 00:00:00.000	4110 Old Redmond Rd.	Redmond	WA	98052	USA	(206) 555-6122	5176	0x151C2F00020000000000E014002100FFFFFFFFFF	2	http://accweb/4	
5	5	Buchanan	Steven	Sales Manager	Mr.	1955-03-03 00:00:00.000	1993-10-17 00:00:00.000	14 Garrett Hill	London	NULL	SW1 8JR	UK	(71) 555-4848	3453	0x151C2F00020000000000E014002100FFFFFFFFFF	2	http://accweb/5	
6	6	Suyama	Michael	Sales Representative	Mr.	1963-07-02 00:00:00.000	1993-10-17 00:00:00.000	Coventry House, Miner Rd.	London	NULL	EC2 7JR	UK	(71) 555-7773	428	0x151C2F00020000000000E014002100FFFFFFFFFF	5	http://accweb/6	
7	7	King	Robert	Sales Representative	Mr.	1960-05-29 00:00:00.000	1994-01-02 00:00:00.000	Edgebury Hollow, Winchester Way	London	NULL	RG1 9SP	UK	(71) 555-5598	465	0x151C2F00020000000000E014002100FFFFFFFFFF	5	http://accweb/7	
8	8	Callahan	Laura	Inside Sales Coordinator	Ms.	1958-01-09 00:00:00.000	1994-03-05 00:00:00.000	4726 - 11th Ave. N.E.	Seattle	WA	98105	USA	(206) 555-1189	2344	0x151C2F00020000000000E014002100FFFFFFFFFF	2	http://accweb/8	
9	9	Dodsworth	Anne	Sales Representative	Ms.	1966-01-27 00:00:00.000	1994-11-15 00:00:00.000	7 Houndstooth Rd.	London	NULL	WG2 7LT	UK	(71) 555-4444	452	0x151C2F00020000000000E014002100FFFFFFFFFF	5	http://accweb/9	

- Steven Buchanan (employeeID = 5) **geeft leiding aan** Anne Dodsworth (employeeID = 2).
- Anne Dodsworth (employeeID = 2) **rappoert aan** Steven Buchanan (employeeID = 5).

We kunnen m.b.v. NIAM zeggen over de relatie tussen de tabel Orders en de tabel Employees:

SQLQuery1.sql - MO... (USR\NUP02 (65)) * MON-4GNDEM3.Northwind - Diagram_1*																																																																																																																																																																																																																																																																																																																																		
-vs8D50.sql - MON-... (USR\NUP02 (56))																																																																																																																																																																																																																																																																																																																																		
-vs772C.sql - MON-... (USR\NUP02 (51))																																																																																																																																																																																																																																																																																																																																		
<pre>SELECT * FROM Employees; SELECT * FROM Orders WHERE EmployeeID = 5;</pre>																																																																																																																																																																																																																																																																																																																																		
<table border="1"> <thead> <tr> <th>EmployeeID</th><th>LastName</th><th>FirstName</th><th>Title</th><th>TitleOfCourtesy</th><th>BirthDate</th><th>HireDate</th><th>Address</th><th>City</th><th>Region</th><th>PostalCode</th><th>Country</th><th>HomePhone</th><th>Extension</th><th>Photo</th><th>Notes</th><th>ReportsTo</th><th>PhotoPath</th></tr> </thead> <tbody> <tr> <td>1</td><td>Davolio</td><td>Nancy</td><td>Sales Representative</td><td>Ms.</td><td>1948-12-08 00:00:00.000</td><td>1992-05-01 00:00:00.000</td><td>507 - 20th Ave. E. Apt. 2A</td><td>Seattle</td><td>WA</td><td>98122</td><td>USA</td><td>(206) 555-9857</td><td>5467</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td><td>2</td><td>http://accweb/1</td></tr> <tr> <td>2</td><td>Fuller</td><td>Andrew</td><td>Vice President, Sales</td><td>Dr.</td><td>1952-02-19 00:00:00.000</td><td>1992-08-14 00:00:00.000</td><td>908 W. Capital Way</td><td>Tacoma</td><td>WA</td><td>98401</td><td>USA</td><td>(206) 555-9482</td><td>3457</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td><td>2</td><td>http://accweb/2</td></tr> <tr> <td>3</td><td>Leverling</td><td>Janet</td><td>Sales Representative</td><td>Ms.</td><td>1963-08-30 00:00:00.000</td><td>1992-04-01 00:00:00.000</td><td>722 Moss Bay Blvd.</td><td>Kirkland</td><td>WA</td><td>98033</td><td>USA</td><td>(206) 555-3412</td><td>3355</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td><td>2</td><td>http://accweb/3</td></tr> <tr> <td>4</td><td>Peacock</td><td>Margaret</td><td>Sales Representative</td><td>Mrs.</td><td>1937-09-19 00:00:00.000</td><td>1993-05-03 00:00:00.000</td><td>4110 Old Redmond Rd.</td><td>Redmond</td><td>WA</td><td>98052</td><td>USA</td><td>(206) 555-6122</td><td>5176</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td><td>2</td><td>http://accweb/4</td></tr> <tr> <td>5</td><td>Buchanan</td><td>Steven</td><td>Sales Manager</td><td>Mr.</td><td>1955-03-04 00:00:00.000</td><td>1993-10-17 00:00:00.000</td><td>14 Garrett Hill</td><td>London</td><td>NULL</td><td>SW1 8JR</td><td>UK</td><td>(71) 555-4848</td><td>3453</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td><td>2</td><td>http://accweb/5</td></tr> <tr> <td>6</td><td>Suyama</td><td>Michael</td><td>Sales Representative</td><td>Mr.</td><td>1963-07-02 00:00:00.000</td><td>1993-10-17 00:00:00.000</td><td>Coventry House, Miner Rd.</td><td>London</td><td>NULL</td><td>EC2 7JR</td><td>UK</td><td>(71) 555-7773</td><td>428</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td><td>5</td><td>http://accweb/6</td></tr> <tr> <td>7</td><td>King</td><td>Robert</td><td>Sales Representative</td><td>Mr.</td><td>1960-05-29 00:00:00.000</td><td>1994-01-02 00:00:00.000</td><td>Edgebury Hollow, Winchester Way</td><td>London</td><td>NULL</td><td>RG1 9SP</td><td>UK</td><td>(71) 555-5598</td><td>465</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td><td>5</td><td>http://accweb/7</td></tr> <tr> <td>8</td><td>Callahan</td><td>Laura</td><td>Inside Sales Coordinator</td><td>Ms.</td><td>1958-01-09 00:00:00.000</td><td>1994-03-05 00:00:00.000</td><td>4726 - 11th Ave. N.E.</td><td>Seattle</td><td>WA</td><td>98105</td><td>USA</td><td>(206) 555-1189</td><td>2344</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td><td>2</td><td>http://accweb/8</td></tr> <tr> <td>9</td><td>Dodsworth</td><td>Anne</td><td>Sales Representative</td><td>Ms.</td><td>1966-01-27 00:00:00.000</td><td>1994-11-15 00:00:00.000</td><td>7 Houndstooth Rd.</td><td>London</td><td>NULL</td><td>WG2 7LT</td><td>UK</td><td>(71) 555-4444</td><td>452</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td><td>5</td><td>http://accweb/9</td></tr> </tbody> </table>																			EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	BirthDate	HireDate	Address	City	Region	PostalCode	Country	HomePhone	Extension	Photo	Notes	ReportsTo	PhotoPath	1	Davolio	Nancy	Sales Representative	Ms.	1948-12-08 00:00:00.000	1992-05-01 00:00:00.000	507 - 20th Ave. E. Apt. 2A	Seattle	WA	98122	USA	(206) 555-9857	5467	0x151C2F00020000000000E014002100FFFFFFFFFF	2	http://accweb/1	2	Fuller	Andrew	Vice President, Sales	Dr.	1952-02-19 00:00:00.000	1992-08-14 00:00:00.000	908 W. Capital Way	Tacoma	WA	98401	USA	(206) 555-9482	3457	0x151C2F00020000000000E014002100FFFFFFFFFF	2	http://accweb/2	3	Leverling	Janet	Sales Representative	Ms.	1963-08-30 00:00:00.000	1992-04-01 00:00:00.000	722 Moss Bay Blvd.	Kirkland	WA	98033	USA	(206) 555-3412	3355	0x151C2F00020000000000E014002100FFFFFFFFFF	2	http://accweb/3	4	Peacock	Margaret	Sales Representative	Mrs.	1937-09-19 00:00:00.000	1993-05-03 00:00:00.000	4110 Old Redmond Rd.	Redmond	WA	98052	USA	(206) 555-6122	5176	0x151C2F00020000000000E014002100FFFFFFFFFF	2	http://accweb/4	5	Buchanan	Steven	Sales Manager	Mr.	1955-03-04 00:00:00.000	1993-10-17 00:00:00.000	14 Garrett Hill	London	NULL	SW1 8JR	UK	(71) 555-4848	3453	0x151C2F00020000000000E014002100FFFFFFFFFF	2	http://accweb/5	6	Suyama	Michael	Sales Representative	Mr.	1963-07-02 00:00:00.000	1993-10-17 00:00:00.000	Coventry House, Miner Rd.	London	NULL	EC2 7JR	UK	(71) 555-7773	428	0x151C2F00020000000000E014002100FFFFFFFFFF	5	http://accweb/6	7	King	Robert	Sales Representative	Mr.	1960-05-29 00:00:00.000	1994-01-02 00:00:00.000	Edgebury Hollow, Winchester Way	London	NULL	RG1 9SP	UK	(71) 555-5598	465	0x151C2F00020000000000E014002100FFFFFFFFFF	5	http://accweb/7	8	Callahan	Laura	Inside Sales Coordinator	Ms.	1958-01-09 00:00:00.000	1994-03-05 00:00:00.000	4726 - 11th Ave. N.E.	Seattle	WA	98105	USA	(206) 555-1189	2344	0x151C2F00020000000000E014002100FFFFFFFFFF	2	http://accweb/8	9	Dodsworth	Anne	Sales Representative	Ms.	1966-01-27 00:00:00.000	1994-11-15 00:00:00.000	7 Houndstooth Rd.	London	NULL	WG2 7LT	UK	(71) 555-4444	452	0x151C2F00020000000000E014002100FFFFFFFFFF	5	http://accweb/9																																																																																																																																					
EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	BirthDate	HireDate	Address	City	Region	PostalCode	Country	HomePhone	Extension	Photo	Notes	ReportsTo	PhotoPath																																																																																																																																																																																																																																																																																																																	
1	Davolio	Nancy	Sales Representative	Ms.	1948-12-08 00:00:00.000	1992-05-01 00:00:00.000	507 - 20th Ave. E. Apt. 2A	Seattle	WA	98122	USA	(206) 555-9857	5467	0x151C2F00020000000000E014002100FFFFFFFFFF	2	http://accweb/1																																																																																																																																																																																																																																																																																																																		
2	Fuller	Andrew	Vice President, Sales	Dr.	1952-02-19 00:00:00.000	1992-08-14 00:00:00.000	908 W. Capital Way	Tacoma	WA	98401	USA	(206) 555-9482	3457	0x151C2F00020000000000E014002100FFFFFFFFFF	2	http://accweb/2																																																																																																																																																																																																																																																																																																																		
3	Leverling	Janet	Sales Representative	Ms.	1963-08-30 00:00:00.000	1992-04-01 00:00:00.000	722 Moss Bay Blvd.	Kirkland	WA	98033	USA	(206) 555-3412	3355	0x151C2F00020000000000E014002100FFFFFFFFFF	2	http://accweb/3																																																																																																																																																																																																																																																																																																																		
4	Peacock	Margaret	Sales Representative	Mrs.	1937-09-19 00:00:00.000	1993-05-03 00:00:00.000	4110 Old Redmond Rd.	Redmond	WA	98052	USA	(206) 555-6122	5176	0x151C2F00020000000000E014002100FFFFFFFFFF	2	http://accweb/4																																																																																																																																																																																																																																																																																																																		
5	Buchanan	Steven	Sales Manager	Mr.	1955-03-04 00:00:00.000	1993-10-17 00:00:00.000	14 Garrett Hill	London	NULL	SW1 8JR	UK	(71) 555-4848	3453	0x151C2F00020000000000E014002100FFFFFFFFFF	2	http://accweb/5																																																																																																																																																																																																																																																																																																																		
6	Suyama	Michael	Sales Representative	Mr.	1963-07-02 00:00:00.000	1993-10-17 00:00:00.000	Coventry House, Miner Rd.	London	NULL	EC2 7JR	UK	(71) 555-7773	428	0x151C2F00020000000000E014002100FFFFFFFFFF	5	http://accweb/6																																																																																																																																																																																																																																																																																																																		
7	King	Robert	Sales Representative	Mr.	1960-05-29 00:00:00.000	1994-01-02 00:00:00.000	Edgebury Hollow, Winchester Way	London	NULL	RG1 9SP	UK	(71) 555-5598	465	0x151C2F00020000000000E014002100FFFFFFFFFF	5	http://accweb/7																																																																																																																																																																																																																																																																																																																		
8	Callahan	Laura	Inside Sales Coordinator	Ms.	1958-01-09 00:00:00.000	1994-03-05 00:00:00.000	4726 - 11th Ave. N.E.	Seattle	WA	98105	USA	(206) 555-1189	2344	0x151C2F00020000000000E014002100FFFFFFFFFF	2	http://accweb/8																																																																																																																																																																																																																																																																																																																		
9	Dodsworth	Anne	Sales Representative	Ms.	1966-01-27 00:00:00.000	1994-11-15 00:00:00.000	7 Houndstooth Rd.	London	NULL	WG2 7LT	UK	(71) 555-4444	452	0x151C2F00020000000000E014002100FFFFFFFFFF	5	http://accweb/9																																																																																																																																																																																																																																																																																																																		
<table border="1"> <thead> <tr> <th>OrderID</th><th>CustomerID</th><th>EmployeeID</th><th>OrderDate</th><th>RequiredDate</th><th>ShippedDate</th><th>Freight</th><th>ShipName</th><th>ShipAddress</th><th>ShipCity</th><th>ShipRegion</th><th>ShipPostalCode</th><th>ShipCountry</th><th>ShipPhone</th><th>ShipFax</th><th>ShipEmail</th><th>ShipRegion</th><th>ShipPostalCode</th><th>ShipCountry</th></tr> </thead> <tbody> <tr> <td>1</td><td>VINET</td><td>5</td><td>1996-07-04 00:00:00.000</td><td>1996-08-01 00:00:00.000</td><td>1996-07-16 00:00:00.000</td><td>3</td><td>23.28</td><td>Vins et alcools Chevalier</td><td>59 rue de l'Abbaye</td><td>Reims</td><td>NULL</td><td>51100</td><td>France</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td><td>5</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td></tr> <tr> <td>2</td><td>CHOPS</td><td>5</td><td>1996-07-11 00:00:00.000</td><td>1996-08-08 00:00:00.000</td><td>1996-07-23 00:00:00.000</td><td>2</td><td>22.38</td><td>Chop-suey Chinese</td><td>Hautpett 31</td><td>Bern</td><td>NULL</td><td>3012</td><td>Switzerland</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td><td>5</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td></tr> <tr> <td>3</td><td>WHITC</td><td>5</td><td>1996-07-31 00:00:00.000</td><td>1996-08-14 00:00:00.000</td><td>1996-08-09 00:00:00.000</td><td>1</td><td>4.56</td><td>White Clover Markets</td><td>1029 - 12th Ave. S.</td><td>Seattle</td><td>WA</td><td>98124</td><td>USA</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td><td>5</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td></tr> <tr> <td>4</td><td>BLONP</td><td>5</td><td>1996-08-04 00:00:00.000</td><td>1996-09-10 00:00:00.000</td><td>1996-09-06 00:00:00.000</td><td>2</td><td>5.74</td><td>Blond père et fils</td><td>24, place Kléber</td><td>Strasbourg</td><td>NULL</td><td>67000</td><td>France</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td><td>5</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td></tr> <tr> <td>5</td><td>WARTH</td><td>5</td><td>1996-10-03 00:00:00.000</td><td>1996-10-17 00:00:00.000</td><td>1996-10-18 00:00:00.000</td><td>3</td><td>34.57</td><td>Wartian Herkku</td><td>Torkku 38</td><td>Oulu</td><td>NULL</td><td>90110</td><td>Finland</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td><td>5</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td></tr> <tr> <td>6</td><td>MAISD</td><td>5</td><td>1996-10-18 00:00:00.000</td><td>1996-10-25 00:00:00.000</td><td>1996-10-25 00:00:00.000</td><td>3</td><td>0.59</td><td>Torikatu 38</td><td>Oulu</td><td>NULL</td><td>90110</td><td>Finland</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td><td>5</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td></tr> <tr> <td>7</td><td>LAMAI</td><td>5</td><td>1996-11-20 00:00:00.000</td><td>1996-12-10 00:00:00.000</td><td>1996-11-27 00:00:00.000</td><td>1</td><td>19.64</td><td>La maison d'Ase</td><td>1 rue Alsace-Lorraine</td><td>Toulouse</td><td>NULL</td><td>31000</td><td>France</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td><td>5</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td></tr> <tr> <td>8</td><td>SEVES</td><td>5</td><td>1996-11-21 00:00:00.000</td><td>1996-12-10 00:00:00.000</td><td>1996-11-26 00:00:00.000</td><td>3</td><td>288.43</td><td>Seven Seas Imports</td><td>90 Vadhurst Rd.</td><td>London</td><td>NULL</td><td>OX15 4ND</td><td>UK</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td><td>5</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td></tr> <tr> <td>9</td><td>QUEEN</td><td>5</td><td>1996-12-04 00:00:00.000</td><td>1997-01-01 00:00:00.000</td><td>1996-12-09 00:00:00.000</td><td>2</td><td>890.78</td><td>Queen Cozinha</td><td>Alameda dos Canais</td><td>Sao Paulo</td><td>SP</td><td>05487-020</td><td>Brazil</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td><td>5</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td></tr> <tr> <td>10</td><td>FOLKO</td><td>5</td><td>1996-12-10 00:00:00.000</td><td>1997-01-07 00:00:00.000</td><td>1996-12-19 00:00:00.000</td><td>3</td><td>5.44</td><td>Folk och fä HB</td><td>Akegatan 24</td><td>Bräcke</td><td>NULL</td><td>S-944 67</td><td>Sweden</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td><td>5</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td></tr> <tr> <td>11</td><td>PRINZ</td><td>5</td><td>1996-12-27 00:00:00.000</td><td>1997-01-24 00:00:00.000</td><td>1997-01-02 00:00:00.000</td><td>1</td><td>60.26</td><td>Princesa Isabel Vinhos</td><td>Estreita da saude</td><td>Lisbon</td><td>NULL</td><td>1756</td><td>Portugal</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td><td>5</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td></tr> <tr> <td>12</td><td>SUPRD</td><td>5</td><td>1997-03-04 00:00:00.000</td><td>1997-03-40 00:00:00.000</td><td>1997-03-06 00:00:00.000</td><td>3</td><td>14.78</td><td>Suprimes délices</td><td>Boulevard Thiers</td><td>Cherfali</td><td>NULL</td><td>B-6000</td><td>Belgium</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td><td>5</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td></tr> <tr> <td>13</td><td>PERIC</td><td>5</td><td>1997-03-13 00:00:00.000</td><td>1997-04-10 00:00:00.000</td><td>1997-03-21 00:00:00.000</td><td>2</td><td>83.49</td><td>Pericos Comidas clási...</td><td>Calle Dr. Jorge C...</td><td>México</td><td>NULL</td><td>05033</td><td>Mexico</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td><td>5</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td></tr> <tr> <td>14</td><td>PRINI</td><td>5</td><td>1997-03-17 00:00:00.000</td><td>1997-04-14 00:00:00.000</td><td>1997-03-25 00:00:00.000</td><td>2</td><td>13.02</td><td>Principe Isabel Vinhos</td><td>Estreita da saude</td><td>Lisboa</td><td>NULL</td><td>1756</td><td>Portugal</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td><td>5</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td></tr> <tr> <td>15</td><td>MAISD</td><td>5</td><td>1997-05-07 00:00:00.000</td><td>1997-05-04 00:00:00.000</td><td>1997-05-09 00:00:00.000</td><td>2</td><td>66.69</td><td>Maison Dewey</td><td>Rue Joseph-Bens...</td><td>Brussels</td><td>NULL</td><td>B-1180</td><td>Belgium</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td><td>5</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td></tr> <tr> <td>16</td><td>QUICK</td><td>5</td><td>1997-05-27 00:00:00.000</td><td>1997-05-10 00:00:00.000</td><td>1997-05-30 00:00:00.000</td><td>1</td><td>171.24</td><td>QUICK-Stop</td><td>Taucherkirche 10</td><td>Curewinkel</td><td>NULL</td><td>01307</td><td>Germany</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td><td>5</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td></tr> <tr> <td>17</td><td>PATTC</td><td>5</td><td>1997-06-16 00:00:00.000</td><td>1997-07-11 00:00:00.000</td><td>1997-07-11 00:00:00.000</td><td>1</td><td>58.98</td><td>Rattenkake Canyon G...</td><td>2871 Milton Dr.</td><td>Abiquer...</td><td>NM</td><td>87110</td><td>USA</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td><td>5</td><td>0x151C2F00020000000000E014002100FFFFFFFFFF</td></tr> <tr> <td>18</td><td>MORGK</td><td>5</td><td>1997-06-20 00:00:00.000</td><td>1997-07-04 00:00:00.000</td><td>1997-06-30 00:00:00.000</td><td>1</td><td>127.34</td><td>Morgenstern Gesund...</td><td>Heerstr. 22</td><td>Lepzig</td><td>NULL</td><td>04179</td><td>Germany</td><td>0x</td></tr></tbody></table>	OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate	ShippedDate	Freight	ShipName	ShipAddress	ShipCity	ShipRegion	ShipPostalCode	ShipCountry	ShipPhone	ShipFax	ShipEmail	ShipRegion	ShipPostalCode	ShipCountry	1	VINET	5	1996-07-04 00:00:00.000	1996-08-01 00:00:00.000	1996-07-16 00:00:00.000	3	23.28	Vins et alcools Chevalier	59 rue de l'Abbaye	Reims	NULL	51100	France	0x151C2F00020000000000E014002100FFFFFFFFFF	5	0x151C2F00020000000000E014002100FFFFFFFFFF	2	CHOPS	5	1996-07-11 00:00:00.000	1996-08-08 00:00:00.000	1996-07-23 00:00:00.000	2	22.38	Chop-suey Chinese	Hautpett 31	Bern	NULL	3012	Switzerland	0x151C2F00020000000000E014002100FFFFFFFFFF	5	0x151C2F00020000000000E014002100FFFFFFFFFF	3	WHITC	5	1996-07-31 00:00:00.000	1996-08-14 00:00:00.000	1996-08-09 00:00:00.000	1	4.56	White Clover Markets	1029 - 12th Ave. S.	Seattle	WA	98124	USA	0x151C2F00020000000000E014002100FFFFFFFFFF	5	0x151C2F00020000000000E014002100FFFFFFFFFF	4	BLONP	5	1996-08-04 00:00:00.000	1996-09-10 00:00:00.000	1996-09-06 00:00:00.000	2	5.74	Blond père et fils	24, place Kléber	Strasbourg	NULL	67000	France	0x151C2F00020000000000E014002100FFFFFFFFFF	5	0x151C2F00020000000000E014002100FFFFFFFFFF	5	WARTH	5	1996-10-03 00:00:00.000	1996-10-17 00:00:00.000	1996-10-18 00:00:00.000	3	34.57	Wartian Herkku	Torkku 38	Oulu	NULL	90110	Finland	0x151C2F00020000000000E014002100FFFFFFFFFF	5	0x151C2F00020000000000E014002100FFFFFFFFFF	6	MAISD	5	1996-10-18 00:00:00.000	1996-10-25 00:00:00.000	1996-10-25 00:00:00.000	3	0.59	Torikatu 38	Oulu	NULL	90110	Finland	0x151C2F00020000000000E014002100FFFFFFFFFF	5	0x151C2F00020000000000E014002100FFFFFFFFFF	7	LAMAI	5	1996-11-20 00:00:00.000	1996-12-10 00:00:00.000	1996-11-27 00:00:00.000	1	19.64	La maison d'Ase	1 rue Alsace-Lorraine	Toulouse	NULL	31000	France	0x151C2F00020000000000E014002100FFFFFFFFFF	5	0x151C2F00020000000000E014002100FFFFFFFFFF	8	SEVES	5	1996-11-21 00:00:00.000	1996-12-10 00:00:00.000	1996-11-26 00:00:00.000	3	288.43	Seven Seas Imports	90 Vadhurst Rd.	London	NULL	OX15 4ND	UK	0x151C2F00020000000000E014002100FFFFFFFFFF	5	0x151C2F00020000000000E014002100FFFFFFFFFF	9	QUEEN	5	1996-12-04 00:00:00.000	1997-01-01 00:00:00.000	1996-12-09 00:00:00.000	2	890.78	Queen Cozinha	Alameda dos Canais	Sao Paulo	SP	05487-020	Brazil	0x151C2F00020000000000E014002100FFFFFFFFFF	5	0x151C2F00020000000000E014002100FFFFFFFFFF	10	FOLKO	5	1996-12-10 00:00:00.000	1997-01-07 00:00:00.000	1996-12-19 00:00:00.000	3	5.44	Folk och fä HB	Akegatan 24	Bräcke	NULL	S-944 67	Sweden	0x151C2F00020000000000E014002100FFFFFFFFFF	5	0x151C2F00020000000000E014002100FFFFFFFFFF	11	PRINZ	5	1996-12-27 00:00:00.000	1997-01-24 00:00:00.000	1997-01-02 00:00:00.000	1	60.26	Princesa Isabel Vinhos	Estreita da saude	Lisbon	NULL	1756	Portugal	0x151C2F00020000000000E014002100FFFFFFFFFF	5	0x151C2F00020000000000E014002100FFFFFFFFFF	12	SUPRD	5	1997-03-04 00:00:00.000	1997-03-40 00:00:00.000	1997-03-06 00:00:00.000	3	14.78	Suprimes délices	Boulevard Thiers	Cherfali	NULL	B-6000	Belgium	0x151C2F00020000000000E014002100FFFFFFFFFF	5	0x151C2F00020000000000E014002100FFFFFFFFFF	13	PERIC	5	1997-03-13 00:00:00.000	1997-04-10 00:00:00.000	1997-03-21 00:00:00.000	2	83.49	Pericos Comidas clási...	Calle Dr. Jorge C...	México	NULL	05033	Mexico	0x151C2F00020000000000E014002100FFFFFFFFFF	5	0x151C2F00020000000000E014002100FFFFFFFFFF	14	PRINI	5	1997-03-17 00:00:00.000	1997-04-14 00:00:00.000	1997-03-25 00:00:00.000	2	13.02	Principe Isabel Vinhos	Estreita da saude	Lisboa	NULL	1756	Portugal	0x151C2F00020000000000E014002100FFFFFFFFFF	5	0x151C2F00020000000000E014002100FFFFFFFFFF	15	MAISD	5	1997-05-07 00:00:00.000	1997-05-04 00:00:00.000	1997-05-09 00:00:00.000	2	66.69	Maison Dewey	Rue Joseph-Bens...	Brussels	NULL	B-1180	Belgium	0x151C2F00020000000000E014002100FFFFFFFFFF	5	0x151C2F00020000000000E014002100FFFFFFFFFF	16	QUICK	5	1997-05-27 00:00:00.000	1997-05-10 00:00:00.000	1997-05-30 00:00:00.000	1	171.24	QUICK-Stop	Taucherkirche 10	Curewinkel	NULL	01307	Germany	0x151C2F00020000000000E014002100FFFFFFFFFF	5	0x151C2F00020000000000E014002100FFFFFFFFFF	17	PATTC	5	1997-06-16 00:00:00.000	1997-07-11 00:00:00.000	1997-07-11 00:00:00.000	1	58.98	Rattenkake Canyon G...	2871 Milton Dr.	Abiquer...	NM	87110	USA	0x151C2F00020000000000E014002100FFFFFFFFFF	5	0x151C2F00020000000000E014002100FFFFFFFFFF	18	MORGK	5	1997-06-20 00:00:00.000	1997-07-04 00:00:00.000	1997-06-30 00:00:00.000	1	127.34	Morgenstern Gesund...	Heerstr. 22	Lepzig	NULL	04179	Germany	0x
OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate	ShippedDate	Freight	ShipName	ShipAddress	ShipCity	ShipRegion	ShipPostalCode	ShipCountry	ShipPhone	ShipFax	ShipEmail	ShipRegion	ShipPostalCode	ShipCountry																																																																																																																																																																																																																																																																																																																
1	VINET	5	1996-07-04 00:00:00.000	1996-08-01 00:00:00.000	1996-07-16 00:00:00.000	3	23.28	Vins et alcools Chevalier	59 rue de l'Abbaye	Reims	NULL	51100	France	0x151C2F00020000000000E014002100FFFFFFFFFF	5	0x151C2F00020000000000E014002100FFFFFFFFFF																																																																																																																																																																																																																																																																																																																		
2	CHOPS	5	1996-07-11 00:00:00.000	1996-08-08 00:00:00.000	1996-07-23 00:00:00.000	2	22.38	Chop-suey Chinese	Hautpett 31	Bern	NULL	3012	Switzerland	0x151C2F00020000000000E014002100FFFFFFFFFF	5	0x151C2F00020000000000E014002100FFFFFFFFFF																																																																																																																																																																																																																																																																																																																		
3	WHITC	5	1996-07-31 00:00:00.000	1996-08-14 00:00:00.000	1996-08-09 00:00:00.000	1	4.56	White Clover Markets	1029 - 12th Ave. S.	Seattle	WA	98124	USA	0x151C2F00020000000000E014002100FFFFFFFFFF	5	0x151C2F00020000000000E014002100FFFFFFFFFF																																																																																																																																																																																																																																																																																																																		
4	BLONP	5	1996-08-04 00:00:00.000	1996-09-10 00:00:00.000	1996-09-06 00:00:00.000	2	5.74	Blond père et fils	24, place Kléber	Strasbourg	NULL	67000	France	0x151C2F00020000000000E014002100FFFFFFFFFF	5	0x151C2F00020000000000E014002100FFFFFFFFFF																																																																																																																																																																																																																																																																																																																		
5	WARTH	5	1996-10-03 00:00:00.000	1996-10-17 00:00:00.000	1996-10-18 00:00:00.000	3	34.57	Wartian Herkku	Torkku 38	Oulu	NULL	90110	Finland	0x151C2F00020000000000E014002100FFFFFFFFFF	5	0x151C2F00020000000000E014002100FFFFFFFFFF																																																																																																																																																																																																																																																																																																																		
6	MAISD	5	1996-10-18 00:00:00.000	1996-10-25 00:00:00.000	1996-10-25 00:00:00.000	3	0.59	Torikatu 38	Oulu	NULL	90110	Finland	0x151C2F00020000000000E014002100FFFFFFFFFF	5	0x151C2F00020000000000E014002100FFFFFFFFFF																																																																																																																																																																																																																																																																																																																			
7	LAMAI	5	1996-11-20 00:00:00.000	1996-12-10 00:00:00.000	1996-11-27 00:00:00.000	1	19.64	La maison d'Ase	1 rue Alsace-Lorraine	Toulouse	NULL	31000	France	0x151C2F00020000000000E014002100FFFFFFFFFF	5	0x151C2F00020000000000E014002100FFFFFFFFFF																																																																																																																																																																																																																																																																																																																		
8	SEVES	5	1996-11-21 00:00:00.000	1996-12-10 00:00:00.000	1996-11-26 00:00:00.000	3	288.43	Seven Seas Imports	90 Vadhurst Rd.	London	NULL	OX15 4ND	UK	0x151C2F00020000000000E014002100FFFFFFFFFF	5	0x151C2F00020000000000E014002100FFFFFFFFFF																																																																																																																																																																																																																																																																																																																		
9	QUEEN	5	1996-12-04 00:00:00.000	1997-01-01 00:00:00.000	1996-12-09 00:00:00.000	2	890.78	Queen Cozinha	Alameda dos Canais	Sao Paulo	SP	05487-020	Brazil	0x151C2F00020000000000E014002100FFFFFFFFFF	5	0x151C2F00020000000000E014002100FFFFFFFFFF																																																																																																																																																																																																																																																																																																																		
10	FOLKO	5	1996-12-10 00:00:00.000	1997-01-07 00:00:00.000	1996-12-19 00:00:00.000	3	5.44	Folk och fä HB	Akegatan 24	Bräcke	NULL	S-944 67	Sweden	0x151C2F00020000000000E014002100FFFFFFFFFF	5	0x151C2F00020000000000E014002100FFFFFFFFFF																																																																																																																																																																																																																																																																																																																		
11	PRINZ	5	1996-12-27 00:00:00.000	1997-01-24 00:00:00.000	1997-01-02 00:00:00.000	1	60.26	Princesa Isabel Vinhos	Estreita da saude	Lisbon	NULL	1756	Portugal	0x151C2F00020000000000E014002100FFFFFFFFFF	5	0x151C2F00020000000000E014002100FFFFFFFFFF																																																																																																																																																																																																																																																																																																																		
12	SUPRD	5	1997-03-04 00:00:00.000	1997-03-40 00:00:00.000	1997-03-06 00:00:00.000	3	14.78	Suprimes délices	Boulevard Thiers	Cherfali	NULL	B-6000	Belgium	0x151C2F00020000000000E014002100FFFFFFFFFF	5	0x151C2F00020000000000E014002100FFFFFFFFFF																																																																																																																																																																																																																																																																																																																		
13	PERIC	5	1997-03-13 00:00:00.000	1997-04-10 00:00:00.000	1997-03-21 00:00:00.000	2	83.49	Pericos Comidas clási...	Calle Dr. Jorge C...	México	NULL	05033	Mexico	0x151C2F00020000000000E014002100FFFFFFFFFF	5	0x151C2F00020000000000E014002100FFFFFFFFFF																																																																																																																																																																																																																																																																																																																		
14	PRINI	5	1997-03-17 00:00:00.000	1997-04-14 00:00:00.000	1997-03-25 00:00:00.000	2	13.02	Principe Isabel Vinhos	Estreita da saude	Lisboa	NULL	1756	Portugal	0x151C2F00020000000000E014002100FFFFFFFFFF	5	0x151C2F00020000000000E014002100FFFFFFFFFF																																																																																																																																																																																																																																																																																																																		
15	MAISD	5	1997-05-07 00:00:00.000	1997-05-04 00:00:00.000	1997-05-09 00:00:00.000	2	66.69	Maison Dewey	Rue Joseph-Bens...	Brussels	NULL	B-1180	Belgium	0x151C2F00020000000000E014002100FFFFFFFFFF	5	0x151C2F00020000000000E014002100FFFFFFFFFF																																																																																																																																																																																																																																																																																																																		
16	QUICK	5	1997-05-27 00:00:00.000	1997-05-10 00:00:00.000	1997-05-30 00:00:00.000	1	171.24	QUICK-Stop	Taucherkirche 10	Curewinkel	NULL	01307	Germany	0x151C2F00020000000000E014002100FFFFFFFFFF	5	0x151C2F00020000000000E014002100FFFFFFFFFF																																																																																																																																																																																																																																																																																																																		
17	PATTC	5	1997-06-16 00:00:00.000	1997-07-11 00:00:00.000	1997-07-11 00:00:00.000	1	58.98	Rattenkake Canyon G...	2871 Milton Dr.	Abiquer...	NM	87110	USA	0x151C2F00020000000000E014002100FFFFFFFFFF	5	0x151C2F00020000000000E014002100FFFFFFFFFF																																																																																																																																																																																																																																																																																																																		
18	MORGK	5	1997-06-20 00:00:00.000	1997-07-04 00:00:00.000	1997-06-30 00:00:00.000	1	127.34	Morgenstern Gesund...	Heerstr. 22	Lepzig	NULL	04179	Germany	0x																																																																																																																																																																																																																																																																																																																				

- Steven Buchanan (employeeID = 5) heeft verkocht order 10358.
- Order 10358 is verkocht door Steven Buchanan (employeeID = 5).

We kunnen zeggen over de relaties tussen de tabellen Orders, Employees en Customers:

**Results**

EmployeeID	Lastname	FirstName	Title	TitleOfCourtesy	BirthDate	HireDate	Address	City	Region	PostalCode	Country	HomePhone	Extension	Photo
1	Davolio	Nancy	Sales Representative	Ms.	1948-12-08 00:00:00.000	1992-05-01 00:00:00.000	507 - 20th Ave. E. Apt. 2A	Seattle	WA	98122	USA	(206) 555-8857	5467	0x151C2F00020000000000E01402100FFFFFFFFFF426
2	Fuller	Andrew	Vice President, Sales	Dr.	1952-02-19 00:00:00.000	1992-08-14 00:00:00.000	908 W Capital Way	Tacoma	WA	98401	USA	(206) 555-9482	3457	0x151C2F00020000000000E01402100FFFFFFFFFF426
3	Leverling	Janet	Sales Representative	Ms.	1963-09-30 00:00:00.000	1992-04-01 00:00:00.000	722 Mess Bay Blvd.	Kirkland	WA	98033	USA	(206) 555-3412	3355	0x151C2F00020000000000E01402100FFFFFFFFFF426
4	Peacock	Margaret	Sales Representative	Mrs.	1937-09-19 00:00:00.000	1993-05-03 00:00:00.000	4110 Old Redmond Rd.	Redmond	WA	98052	USA	(206) 555-8122	5176	0x151C2F00020000000000E01402100FFFFFFFFFF426
5	Buchanan	Steven	Sales Manager	Mr.	1955-03-04 00:00:00.000	1993-10-17 00:00:00.000	14 Garrett Hill	London	NULL	SW1 8JR	UK	(71) 555-4484	3453	0x151C2F00020000000000E01402100FFFFFFFFFF426
6	Supama	Michael	Sales Representative	Mr.	1963-07-02 00:00:00.000	1993-10-17 00:00:00.000	Covertry House, Miner Rd.	London	NULL	EC2 7JR	UK	(71) 555-7773	428	0x151C2F00020000000000E01402100FFFFFFFFFF426
7	King	Robert	Sales Representative	Mr.	1965-05-29 00:00:00.000	1994-01-02 00:00:00.000	Edgeham Hollow, Winchester Way	London	NULL	RG1 9SP	UK	(71) 555-5588	465	0x151C2F00020000000000E01402100FFFFFFFFFF426
8	Callahan	Laura	Inside Sales Coordinator	Ms.	1958-01-09 00:00:00.000	1994-03-05 00:00:00.000	4726 - 11th Ave. N.E.	Seattle	WA	98105	USA	(206) 555-1189	2344	0x151C2F00020000000000E01402100FFFFFFFFFF426
9	Dodsworth	Anne	Sales Representative	Ms.	1968-01-27 00:00:00.000	1994-11-15 00:00:00.000	7 Hounds tooth Rd.	London	NULL	WG2 7LT	UK	(71) 555-4444	452	0x151C2F00020000000000E01402100FFFFFFFFFF426

OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate	ShippedDate	ShipVia	Freight	ShipName	ShipAddress	ShipCity	ShipRegion	ShipPostalCode	ShipCountry
106	10353	PICCO	7	1996-11-13 00:00:00.000	1996-11-25 00:00:00.000	3	360.53	Piccolo und mehr	Gesweg 14	Salzburg	NULL	5020	Austria
107	10354	PERIC	8	1996-11-14 00:00:00.000	1996-12-12 00:00:00.000	3	53.80	Pericles Comidas clásicas	Calle Dr. Jorge Cañiz 321	Mexico D.F.	NULL	05033	Mexico
108	10358	AROUT	6	1996-11-15 00:00:00.000	1996-12-13 00:00:00.000	1	41.95	Around the Horn	Brook Farm Stratford St. Mary	Colchester	Essex	C07 SJX	UK
109	10359	WANDS	6	1996-11-18 00:00:00.000	1996-12-16 00:00:00.000	1	36.71	Die Wanderinge Kuh	Adenauerallee 900	Stuttgart	NULL	70563	Germany
110	10361	LILAS	3	1996-11-19 00:00:00.000	1996-12-27 00:00:00.000	3	34.88	LILA-Supermercado	Carrera 52 con Ave. Bolívar #65-38 Llano	Barquisimeto	Lara	3508	Venezuela
111	10368	LAMAI	5	1996-11-20 00:00:00.000	1996-12-18 00:00:00.000	1	19.64	La maison d'Asie	1 rue Alsace-Lorraine	Toulouse	NULL	31000	France
112	10359	TES	5	1996-11-21 00:00:00.000	1996-12-19 00:00:00.000	3	288.43	Seven Seas Imports	90 Wadhurst Rd.	London	NULL	OX14 4NB	UK
113	10360	EONP	4	1996-11-22 00:00:00.000	1996-12-20 00:00:00.000	3	131.70	Bondu pêche et fils	24 place Kléber	Straßburg	NULL	67000	France
114	10361	QUICK	1	1996-11-22 00:00:00.000	1996-12-07 00:00:00.000	2	183.17	QUICK-Stop	Tauchentrale 10	Cunewalde	NULL	01307	Germany
115	10362	BONAP	3	1996-11-25 00:00:00.000	1996-12-23 00:00:00.000	1	96.04	Bon app	12, rue des Bouchers	Marseille	NULL	13008	France

CustomerID	CompanyName	ContactName	ContactTitle	Address	City	Region	PostalCode	Country	Phone	Fax	
37	HUNGO	Hungry Owl All-Night Groceries	Patricia McKenna	Sales Associate	8 Johnstown Road	Cork	Co. C...	Ireland	2967542	29673333	
38	ISLAT	Island Trading	Helen Bennett	Marketing Manager	Garden Courtway	Cowes	Ile o...	UK	(198) 555-8888	NULL	
39	KOENE	Königlich Essen	Philip Cramer	Sales Associate	Maubeuge	90	Brandenb...	Germany	0555-09876	NULL	
40	LACFR	La come d'abondance	Daniel Torini	Sales Representative	67, avenue de l'Europe	Versailles	78000	France	03 59 84 10	30 59 85 11	
41	LAMAI	Le maison d'Asie	Annette Roulet	Sales Manager	1 rue Alsace-Lorraine	Toulouse	NULL	France	31000	France	
42	LAUGB	Laughing Bacchus Wine Cellars	Yoshi Tannamuri	Marketing Assistant	1900 Oak St.	Vancouver	BC	Canada	(604) 555-3392	(604) 555-7293	
43	LAZYK	Lazy K Kountry Store	John Steel	Marketing Manager	12 Orchid Terrace	Walla Walla	WA	USA	(509) 555-7986	(509) 555-6221	
44	LEHMS	Lehmanns Markstrand	Renate Messner	Sales Representative	Magazinweg 7	Frankfurt ...	60528	Germany	069-0245984	069-0245874	
45	LETSS	Let's Stop N Shop	Jane Yornes	Owner	87 Polk St. Suite 5	San Franc...	CA	USA	(415) 555-5938	NULL	
46	LILAS	LILA-Supermercado	Carlos González	Accounting Manager	Carrera 52 con Ave. Bolívar...	Bogotá...	Lara	Venez...	3508	(933)-1954	(933)-7256

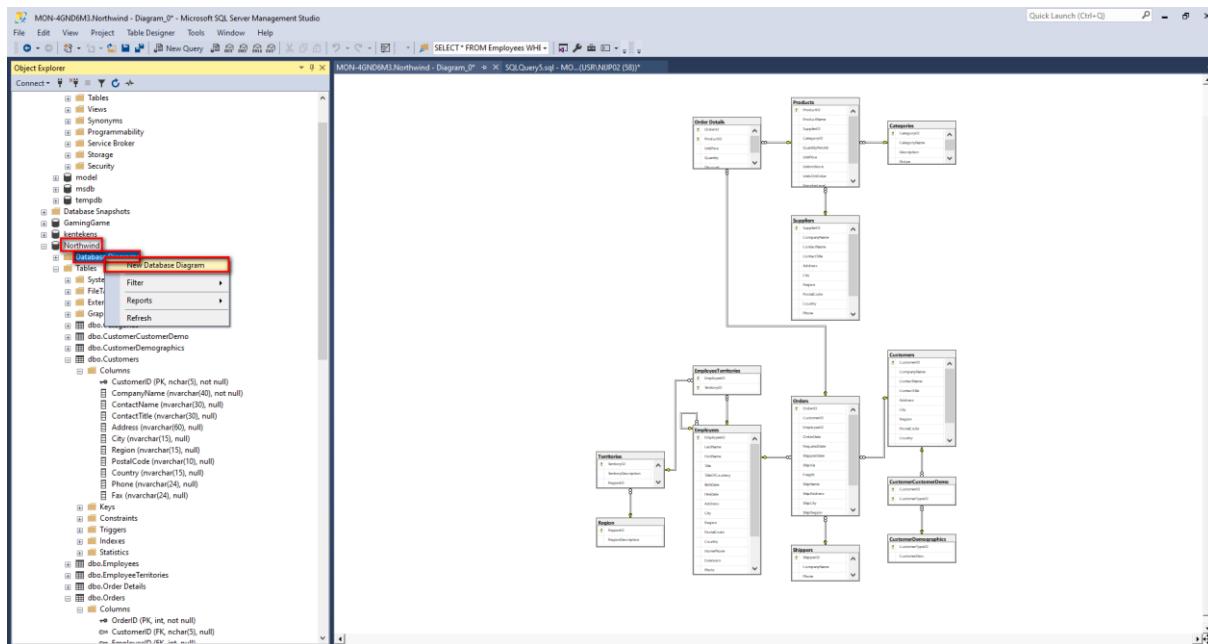
- Order 10358 is besteld door La maison d'Asie en is verkocht door Steven Buchanan.

Conclusie van bovenstaande is dat we de gewenste informatie kunnen verzamelen door gegevens uit verschillende tabellen te combineren en waarbij de gebruikte relatiekolommen (hierboven EmployeeID = 5 en CustomerID = LAMAI) zelf niet in het resultaat zit.

### 6.3 Northwind datamodel in SSMS

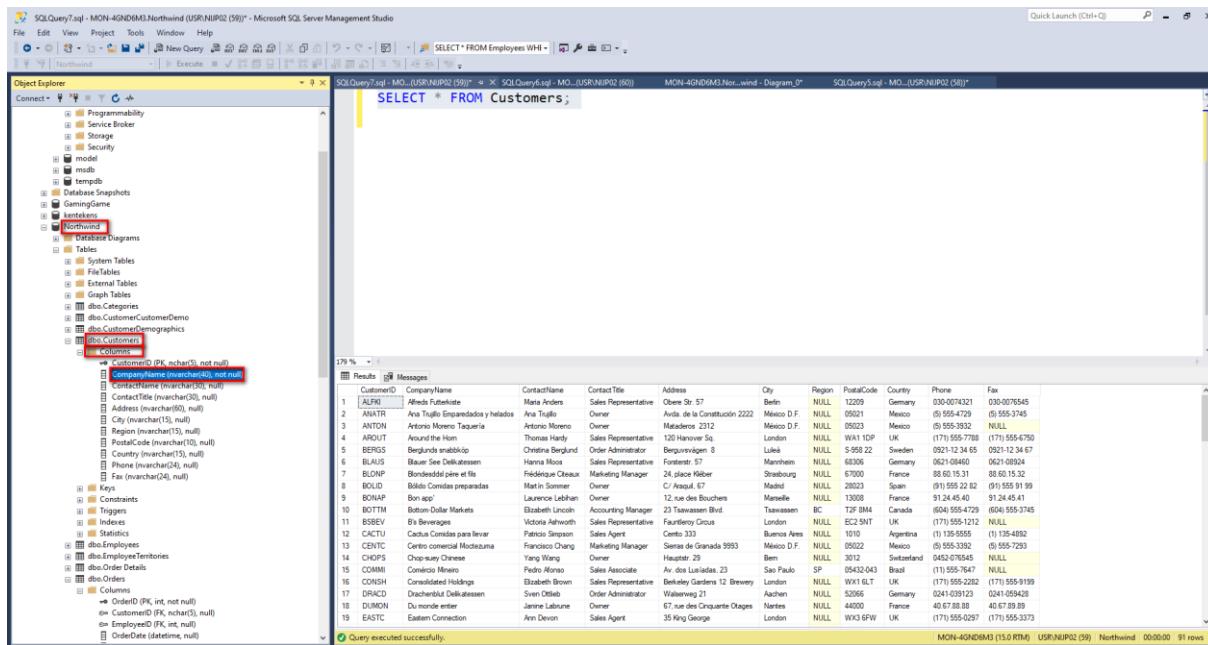
In SSMS kunnen we de tabellen in de database Northwind bekijken en bijvoorbeeld de kolomnamen binnen de tabellen opzoeken.

Zoals beschreven in paragraaf 5.6 kunnen we in SSMS een Entity Relation Diagram genereren van de Northwind database:



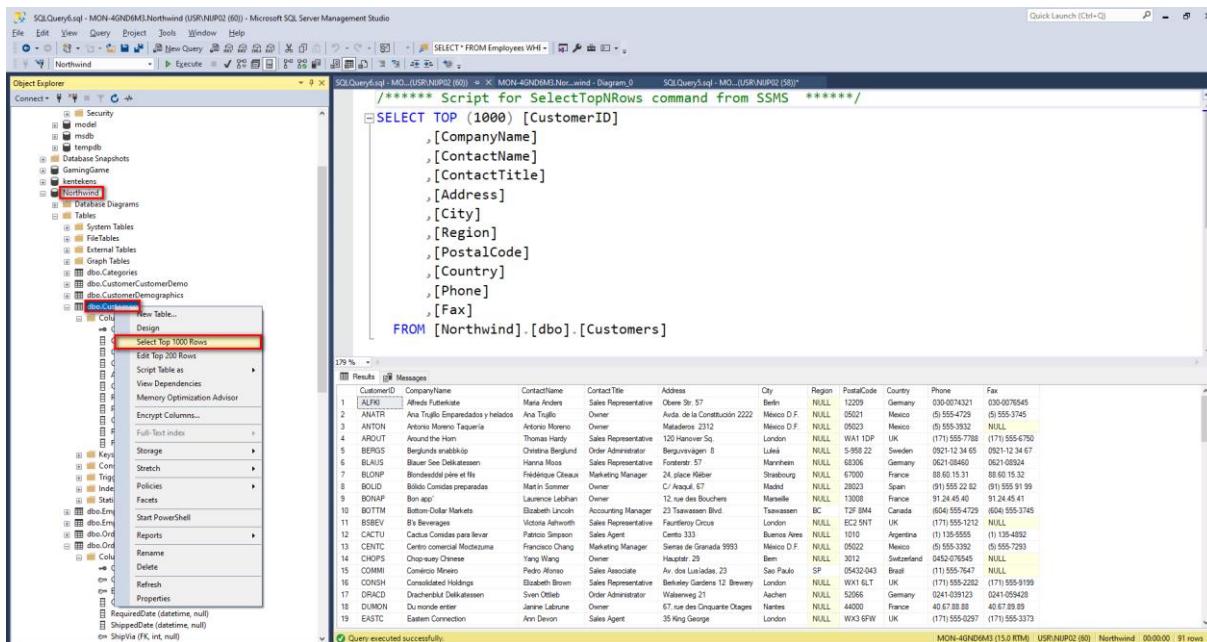
Meestal is het gegenereerde diagram grafisch dan nog niet helemaal netjes en moeten de blokken een beetje verschoven worden om het overzichtelijk te maken.

Zoals beschreven in paragrafen 5.3 en 5.4 kunnen we de tabellen en kolommen in SSMS zichtbaar maken:



CustomerID	CompanyName	ContactName	ContactTitle	Address	City	Region	PostalCode	Country	Phone	Fax
ALFKI	Aleph Furniture	Maria Anders	Sales Representative	Oberstraße 57	Berlin	NULL	12209	Germany	(030) 274321	(030) 270545
ANATR	Anatalia Traiteur y helados	Ana Trujillo	Owner	Avenida de la Constitución 2222	México D.F.	NULL	06521	Mexico	(555) 4729	(555) 3745
ANTON	Antonio Moreno Taquería	Antonio Moreno	Owner	Mataderos 2312	México D.F.	NULL	05023	Mexico	(555) 3932	NULL
ARDUO	Around the Horn	Thomas Hardy	Sales Representative	120 Hanover Sq.	London	NULL	WA1 1OP	UK	(171) 555-7788	(171) 555-6750
BERGS	Berglunds snabbköp	Christina Berglund	Order Administrator	Berguvsgatan 8	Luleå	NULL	S-955 22	Sweden	(092) 12 34 65	(092) 12 34 67
BLAUS	Bauer See Delikatessen	Hanna Moos	Sales Representative	Fonterre St. 43	Mannheim	NULL	68306	Germany	(0621) 08460	(0621) 09582
BLOND	Boscombe Pâté et Fils	Rodney Creek	Marketing Manager	24, rue de Sèvres	Strasbourg	NULL	67000	France	(03) 88 25 47	(03) 88 25 48
BONAP	Bon app'	Laurence Lebihan	Owner	12, rue des Bouches	Manosque	NULL	13000	France	(31) 55 22 82	(31) 55 21 99
BOTTM	Bottom-Dollar Markets	Elizabeth Lincoln	Accounting Manager	23 Taewanen Blvd	Toronto	BC	T2F 1M4	Canada	(604) 555-4729	(604) 555-3745
BSBEV	B's Beverages	Victoria Abbott	Sales Representative	Fauntley Circus	London	NULL	EC2 5NT	UK	(171) 555-1215	NULL
CACTU	Cactus Comidas para llevar	Patricia Simpson	Sales Agent	Cento 333	Buenos Aires	NULL	1010	Argentina	(1) 135-5555	(1) 138-4982
CENTC	Centro comercial Móvil	Franisco Chang	Marketing Manager	Sieras de Granada 9993	México D.F.	NULL	05022	Mexico	(555) 3392	(555) 7293
CHILS	Chile Peperama	Yuri Lomax	Owner	90 rue des Rosiers	Montreal	QC	H3B 2X2	Canada	(514) 871-2323	(514) 871-2322
COMMI	Comisaria Meiggs	Pedro Alonso	Sales Associate	Ave. dos Lázares 23	Sao Paulo	SP	05423-043	Brazil	(11) 555-7647	NULL
CONSH	Consolidated Holdings	Elizabeth Brown	Sales Representative	Bentley Gardens 23	London	NULL	W1K 6LT	UK	(171) 555-2282	(171) 555-9199
DRACD	Drachenblut Delikatessen	Sven Ottieb	Order Administrator	Walzenweg 21	Aachen	NULL	52066	Germany	(0241) 039123	(0241) 059428
DUMON	Du monde entier	Jeanne Labrune	Owner	67, rue des Quatre Otages	Nantes	NULL	44000	France	40 67 88 88	40 67 89 89
EASTC	Eastern Connection	Ann Devon	Sales Agent	35 King George	London	NULL	WX3 6FW	UK	(171) 555-0297	(171) 555-3373

En zoals beschreven in paragraaf 5.5 kunnen we de gegevens in de tabel tonen:



The screenshot shows the SSMS interface with the Northwind database selected. In the Object Explorer, the 'Customers' table under the 'Northwind' schema is selected. A context menu is open over the table, with the option 'Select Top 1000 Rows' highlighted.

The results pane displays the top 91 rows of the 'Customers' table:

CustomerID	CompanyName	ContactName	ContactTitle	Address	City	Region	PostalCode	Country	Phone	Fax	
ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative	Obere Str. 57	Berlin	NLL	12209	Germany	(030) 0074321	(030) 0076545	
ANATR	Ava Trujillo Emparedados y Helados	Ana Trujillo	Owner	Avda. de la Constitución 2222	México D.F.	NLL	05021	Mexico	(5) 555-4729	(5) 555-3745	
ANTON	Antonio Moreno Taquería	Antonio Moreno	Owner	Mateleños 2312	México D.F.	NLL	05023	Mexico	(5) 555-3932	NLL	
AROUT	Around the Horn	Thomas Hardy	Sales Representative	120 Hailsham Rd.	London	NLL	SW1A 1OP	UK	(171) 555-0234	(171) 555-0750	
BERGS	Berglunds cschukip	Christina Berglund	Sales Representative	Ångströmsgatan 9	Stockholm	NLL	11386	Sweden	(08) 555-34 05	(08) 555-34 07	
BLAUS	Bauerp See Delikassen	Hanna Moos	Sales Representative	Fonseca 8	Mannheim	NLL	62036	Germany	(061) 08460	(061) 08924	
BLONP	Blondel père et fils	Fédérico Ocheau	Marketing Manager	24, place Reiber	Strasbourg	NLL	67000	France	88 60 15 31	88 60 15 32	
BOLID	Bólido Comidas preparadas	Martín Sommer	Owner	C/ Aragó, 67	Madrid	NLL	28023	Spain	(91) 555-22 62	(91) 555-91 99	
BONAP	Bon app'	Laurence Lebihan	Owner	12, rue des Bouchers	Marseille	NLL	13008	France	91 24 45 40	91 24 45 41	
BOTTM	Bottom-Dollar Markets	Elizabeth Lincoln	Accounting Manager	23 Tavassey Blvd.	Toronto	T2F 8M4	Canada	(604) 555-4720	(604) 555-3740		
BURGEV	Burgers "R" Us	Victoria Atwood	Sales Representative	2004 Half-Shell Circus	London	EC2 5NT	NLL	UK	(171) 555-0234	(171) 555-0235	
CACOI	Cactus Comidas para llevar	Peregrina Estevez	Sales Agent	Carrer 333	Buenos Aires	NLL	1100	Argentina	(11) 555-6588	(11) 555-4982	
CENTC	Centro comercial ModaZona	François Chang	Marketing Manager	Serrano de Granada 9993	México D.F.	NLL	05022	Mexico	(5) 555-3920	(5) 555-7293	
CHOPS	Chop-suey Chinese	Yang Wang	Owner	Hauptstr. 29	Bern	NLL	3012	Switzerland	(043) 076545	NLL	
COMMI	Comerio Mivaro	Pedro Alonso	Sales Associate	Av. dos Lusíadas, 23	San Paolo	SP	05432-043	Brasil	(11) 555-7647	NLL	
CONSH	Consolidated Holdings	Elizabeth Brown	Sales Representative	Berkley Gardens 12	Brewery	London	NLL	WX1 6LT	UK	(171) 555-2236	(171) 555-9199
DRACD	Drachenblut Delikatessen	Sven Ottie	Order Administrator	Walewweg 21	Aachen	NLL	52056	Germany	(0241) 039123	(0241) 059428	
DUMON	Du monde entier	Jannie Labuhn	Owner	67, rue des Cinq-Sept Ogives	Nantes	NLL	44000	France	40 67 88 88	40 67 89 89	
EASTC	Eastern Connection	Ann Devon	Sales Agent	35 King George	London	NLL	W1K 6FW	UK	(171) 555-0267	(171) 555-3373	

## 6.4 Factuur gegenereerd uit database

Een factuur of bestelling (Order) van Northwind Traders kan er als volgt uitzien nadat het uit de database is gegenereerd:

Northwind Traders  
3066 Long Rd, Green Lane  
Pennsylvania  
18054  
United States

LILA-Supermercado  
attn. Carlos González  
Carrera 52 con Ave.  
Barquisimeto  
3508  
Venezuela

Pennsylvania, 2022-07-10

INVOICE  
OrderID : 11071  
Sales Employee : Nancy Davolio

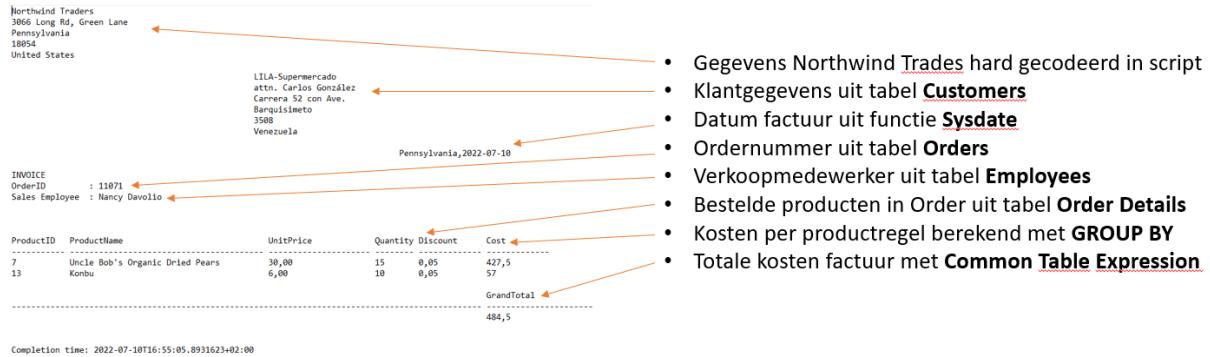
ProductID	ProductName	UnitPrice	Quantity	Discount	Cost
7	Uncle Bob's Organic Dried Pears	30,00	15	0,05	427,5
13	Konbu	6,00	10	0,05	57
GrandTotal					484,5

Completion time: 2022-07-10T16:55:05.8931623+02:00

Deze factuur is volledig uit de database gegenereerd zonder dat een persoon er handmatig werk aan hoeft te verrichten. Het script dat voor het genereren van dit rapport is gebruikt is te vinden in bijlage 3. Je hoeft het script nu nog niet volledig te begrijpen, maar je kan het om te testen kopiëren en plakken naar SSMS en uitvoeren binnen de Northwind database.

Waar het nu om gaat is dat je herkent dat in de factuur gegevens staan die uit verschillende Northwind tabellen zijn gehaald.

## Northwind Trades factuur



The diagram illustrates the components of a Northwind facture:

- Gegevens Northwind Trades hard gecodeerd in script
- Klantgegevens uit tabel **Customers**
- Datum factuur uit functie **Sysdate**
- Ordernummer uit tabel **Orders**
- Verkoopmedewerker uit tabel **Employees**
- Bestelde producten in Order uit tabel **Order Details**
- Kosten per productregel berekend met **GROUP BY**
- Totale kosten factuur met **Common Table Expression**

**Customer Information:**

Northwind Traders	3066 Long Rd, Green Lane
Pennsylvania	
18054	
United States	

**Customer Address:**

LILA Supermercado	
attn: Carlos González	
Carrera 52 con Ave.	
Barquisimeto	
3508	
Venezuela	

**Invoice Details:**

INVOICE	
OrderID : 11071	
Sales Employee : Nancy Davolio	

**Order Details:**

ProductID	ProductName	UnitPrice	Quantity	Discount	Cost
7	Uncle Bob's Organic Dried Pears	30,00	15	0,05	427,5
13	Konbu	6,00	10	0,05	57
					484,5
					GrandTotal

**Completion time:** 2022-07-10T16:55:05.8931623+02:00

Het oplossen van SQL vraagstukken betekent telkens weer puzzelen welke gegevens je uit welke tabellen, welke relaties en welke database functies je moet gebruiken om het rapport (b.v. factuur) samen te kunnen stellen. Daarvoor is het noodzakelijk dat je weet hoe je de databasetaal SQL moet gebruiken en heeft het de voorkeur dat je wegwijs bent in het datadiagram van de Northwind Trades database.

## Hoofdstuk 7 SELECT statement

### 7.1 SELECT FROM

We kunnen gegevens uit een database opvragen door het gebruik van een **query**. Query is het Engelse woord voor vraagstelling. Het is een term die gebruikt wordt binnen de informatica om een opdracht aan een database te beschrijven waarbij een actie in gang wordt gezet. Daarnaast is het de term voor een zoekopdracht zoals deze in een zoekmachine wordt ingevoerd.

M.b.v. het SELECT statement kunnen we gegevens uit een tabel opvragen en als resultaat op het scherm of in een rapport weergeven. Een SELECT query wordt syntactisch geschreven als:

**SELECT <kolom1>, <kolom2>,... FROM <tabel>;**

Achter SELECT geven we aan welke kolommen we uit de tabel willen selecteren. Achter FROM geven we aan uit welke tabel we de gegevens willen opvragen. Elke query wordt afgesloten met een ; - teken.

Het \*-teken betekent dat je 'alle' kolommen uit de tabel wilt selecteren.

**SELECT \* FROM <tabel>;**

Voorbeeld:

Stel we hebben een tabel met de naam StudentenInKlassen:

Studentnummer	Voorletters	Achternaam	Geslacht	geboortedatum	Adres	Huisnummer	Postcode	Woonplaats	telefoonnummer	Klas	Locatie	Adres	Postcode	Plaats	Klassedocent
S0001	A.	Baan	M	12-5-2004	Terwestenstraat	18	2525GH	Den Haag	06-12345678	1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P.Jansen
S0002	S.	Hijmans	V	7-12-2004	Swelinkplein	16	2517GM	Den Haag	06-23456789	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0003	J.	Jansen	M	14-3-2004	Brinkershof	17	2713TX	Zoetermeer	071-1234567	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
S0004	J.	Jansen	M	14-3-2004	Brinkershof	17	2713TX	Zoetermeer	071-1234567	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
S0005	A.	Maarsen	M	12-5-2004	Rembrandtstraat	2	2526PZ	Den Haag	06-56789012	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
S0006	K.	Jakobs	V	12-12-2005	De Vliegerstraat	55	2525BG	Den Haag	06-67890123	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0007	A.B.	Willemsen	M	14-5-2004	Terwestenstraat	16	2525GH	Den Haag	06-78901234	1B	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P.Jansen
S0008	U.	Jans	M	17-6-2005	Herman Costerstraat	162	2571PD	Den Haag	06-89012345	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
S0009	S.	Huber	M	12-5-2004	Hobbemstraat	22	2526JP	Den Haag	06-90123456	1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P.Jansen
S0010	Z.	Wessels	V	5-8-2004	Retiefstraat	29	2571PS	Den Haag	06-13579135	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0011	A.	Kassenberg	M	12-5-2006	Almeloplein	12	2533AA	Den Haag	06-24680246	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
S0012	L.	Vrogers	M	12-5-2004	van der Vennestraat	85	2525CC	Den Haag	06-35791357	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
S0013	A.D.	Eifering	M	5-9-2005	Reitstraat	52	2571RT	Den Haag	06-57913579	1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P.Jansen
S0014	I.	Sebastiaans	M	12-5-2004	Rembrandtstraat	1	2526PN	Den Haag	06-79135791	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
S0015	J.	de Jong	M	12-12-2006	Miereveldstraat	62	2525EG	Den Haag	06-91357913	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
S0016	T.	Boeren	V	12-8-2004	Herman Costerstraat	164	2571PD	Den Haag	06-13456789	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0017	J.	Huijzinga	M	12-5-2005	Terwestenstraat	20	2525GH	Den Haag	06-25678901	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
S0018	D.	Pol	M	12-12-2004	Wesselsstraat	283	2572RZ	Den Haag		1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P.Jansen
S0019	E.	Reitsema	M	12-5-2004	Retiefstraat	27	2571PS	Den Haag	06-47890123	1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P.Jansen
S0020	P.	Vos	M	5-3-2004	van Miereveldstraat	64	2525EG	Den Haag	06-69012345	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0021	K.	Hazenberg	V	12-5-2004	van der Vennestraat	83	2525CC	Den Haag	06-71234567	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
S0022	L.	Joeman	M	12-5-2005	Hobbemstraat	20	2526JP	Den Haag	06-94567890	1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P.Jansen
S0023	S.	Claasen	-	8-9-2004	Terwestenstraat	111	2525GG	Den Haag	06-10987654	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
S0024	W.	de Wit	M	12-5-2004	Almeloplein	10	2533AA	Den Haag	06-29876543	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
S0025	O.	Snellers	M	12-10-2005	Herman Costerstraat	160	2571PD	Den Haag	06-30986543	1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P.Jansen
S0026	H.	Blokhuis	V	8-10-2006	Rembrandtstraat	3	2526PN	Den Haag	06-41234098	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
S0027	D.	Haverkamp	V	12-5-2004	Retiefstraat	25	2571PS	Den Haag	06-52345679	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0028	B.	Boi	M	12-12-2006	AAAAAAA	22	2525GH	Den Haag	06-59873456	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
S0029	K.	Reeuwenkamp	M	22-11-2005	Reitstraat	50	2571RT	Den Haag	06-59873112	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
S0030	L.	van den Broek	M	12-5-2004	Wesselsstraat	281	2572RZ	Den Haag	06-22095586	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen

Wanneer we uitvoeren:

**SELECT \* FROM StudentenInKlassen;**

Dan krijgen we als resultaat alle gegevens uit de tabel StudentenInKlassen:

Studentnummer	Voorletters	Achternaam	Geslacht	Geboortedatum	Adres	Huisnummer	Postcode	Woonplaats	telefoonnummer	Klas	Locatie	Adres	Postcode	Plaats	Klassedocent
S0001	A.	Baan	M	12-5-2004	Terwestenstraat	18	2525GH	Den Haag	06-12345678	1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P.Jansen
S0002	S.	Hijmans	V	7-12-2004	Sweelinckplein	16	2517GM	Den Haag	06-23456789	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0003	J.	Jansen	M	14-3-2004	Brinkershof	17	2713TX	Zoetermeer	071-1234567	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
S0004	J.	Jansen	M	14-3-2004	Brinkershof	17	2713TX	Zoetermeer	071-1234567	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
S0005	A.	Maarsen	M	12-5-2004	Rembrandtstraat	2	2526PZ	Den Haag	06-56789012	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Stooten
S0006	K.	Jakobs	V	12-12-2005	De Vliegerstraat	55	2525BG	Den Haag	06-67890123	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0007	A.B.	Willemsen	M	14-5-2004	Terwestenstraat	16	2525GH	Den Haag	06-78901234	1B	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P.Jansen
S0008	U.	Jans	M	17-6-2005	Herman Costerstraat	162	2571PD	Den Haag	06-89012345	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
S0009	S.	Huber	M	12-5-2004	Hobbemastraat	22	2526JP	Den Haag	06-90123456	1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P.Jansen
S0010	Z.	Wessels	V	5-8-2004	Retiefstraat	29	2571PS	Den Haag	06-13579135	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0011	A.	Kassenberg	M	12-5-2006	Almeloplein	12	2533AA	Den Haag	06-24680246	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Stooten
S0012	L.	Vrogers	M	12-5-2004	van der Vennestraat	85	2525CC	Den Haag	06-55791357	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
S0013	A.D.	Eifering	M	5-9-2005	Reitstraat	52	2571RT	Den Haag	06-57913579	1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P.Jansen
S0014	I.	Sebastiaans	M	12-5-2004	Rembrandtstraat	1	2526PN	Den Haag	06-79135791	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
S0015	J.	de Jong	M	12-12-2006	Miereveldstraat	62	2525EG	Den Haag	06-91357913	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Stooten
S0016	T.	Boeren	V	12-8-2004	Herman Costerstraat	164	2571PD	Den Haag	06-13456789	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0017	J.	Huizinga	M	12-5-2005	Terwestenstraat	20	2525GH	Den Haag	06-25678901	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Stooten
S0018	D.	Pol	M	12-12-2004	Wesselstraat	283	2572RZ	Den Haag		1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P.Jansen
S0019	E.	Reitsema	M	12-5-2004	Retiefstraat	27	2571PS	Den Haag	06-47890123	1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P.Jansen
S0020	P.	Vos	M	5-3-2004	van Miereveldstraat	64	2525EG	Den Haag	06-69012345	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0021	K.	Hazenberg	V	12-5-2004	van der Vennestraat	83	2525CC	Den Haag	06-71234567	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Stooten
S0022	L.	Joemman	M	12-5-2005	Hobbemastraat	20	2526JP	Den Haag	06-945567890	1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P.Jansen
S0023	S.	Claasen	-	8-9-2004	Terwestenstraat	111	2525GG	Den Haag	06-10987654	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
S0024	W.	de Wit	M	12-5-2004	Almeloplein	10	2533AA	Den Haag	06-29876543	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Stooten
S0025	O.	Snellers	M	12-10-2005	Herman Costerstraat	160	2571PD	Den Haag	06-30986543	1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P.Jansen
S0026	H.	Blokhus	V	8-10-2006	Rembrandtstraat	3	2526PN	Den Haag	06-41234098	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Stooten
S0026	D.	Haverkamp	V	12-5-2004	Retiefstraat	25	2571PS	Den Haag	06-52345679	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0026	B.	Bol	M	12-12-2006	AAAAAAAAAAAAAA	22	2525GH	Den Haag	06-59873456	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Stooten
S0027	K.	Reuvenkamp	M	22-11-2005	Reitstraat	50	2571RT	Den Haag	06-59873112	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Stooten
S0028	L.	van den Broek	M	12-5-2004	Wesseisstraat	281	2572RZ	Den Haag	06-22095586	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen

We kunnen alle kolommen i.p.v. \* ook selecteren door alle kolommen expliciet te benoemen:

**SELECT Studentnummer, Voorletters, Achternaam, Geslacht, Geboortedatum, Adres, Huisnummer, Postcode, Woonplaats, Telefoonnummer, Klas, Locatie, Adres, Postcode, Plaats, Klassedocent FROM StudentenInKlassen;**

We kunnen ook slechts een deel van de kolommen uit de tabel selecteren. Wanneer we uitvoeren:

**SELECT Studentnummer, Voorletters, Achternaam FROM StudentenInKlassen;**

Dan krijgen we als resultaat alleen de gegevens uit de kolommen Studentnummer, Voorletters en Achternaam van de tabel StudentenInKlassen:

Studentnummer	Voorletters	Achternaam
S0001	A.	Baan
S0002	S.	Hijmans
S0003	J.	Jansen
S0004	J.	Jansen
S0005	A.	Maarsen
S0006	K.	Jakobs
S0007	A.B.	Willemsen
S0008	U.	Jans
S0009	S.	Huber
S0010	Z.	Wessels
S0011	A.	Kassenberg
S0012	L.	Vrogers
S0013	A.D.	Eifering
S0014	I.	Sebastiaans
S0015	J.	de Jong
S0016	T.	Boeren
S0017	J.	Huizinga
S0018	D.	Pol
S0019	E.	Reitsema
S0020	P.	Vos
S0021	K.	Hazenberg
S0022	L.	Joemman
S0023	S.	Claasen
S0024	W.	de Wit
S0025	O.	Snellers
S0026	H.	Blokhus
S0026	D.	Haverkamp
S0026	B.	Bol
S0027	K.	Reuvenkamp
S0028	L.	van den Broek

Wanneer we uitvoeren:

**SELECT Studentnummer, Klas, Locatie FROM StudentenInKlassen;**

Dan krijgen we als resultaat alleen de gegevens uit de kolommen Studentnummer, Klas en Locatie van de tabel StudentenInKlassen:

Studentnummer	Klas	Locatie
S0001	1A	Gebouw H
S0002	1B	Gebouw R
S0003	1C	Gebouw Z
S0004	1C	Gebouw Z
S0005	1D	Gebouw D
S0006	1B	Gebouw R
S0007	1B	Gebouw H
S0008	1C	Gebouw Z
S0009	1A	Gebouw H
S0010	1B	Gebouw R
S0011	1D	Gebouw D
S0012	1C	Gebouw Z
S0013	1A	Gebouw H
S0014	1C	Gebouw Z
S0015	1D	Gebouw D
S0016	1B	Gebouw R
S0017	1D	Gebouw D
S0018	1A	Gebouw H
S0019	1A	Gebouw H
S0020	1B	Gebouw R
S0021	1D	Gebouw D
S0022	1A	Gebouw H
S0023	1C	Gebouw Z
S0024	1D	Gebouw D
S0025	1A	Gebouw H
	1D	Gebouw D
S0026	1B	Gebouw R
<b>S0026</b>	1D	Gebouw D
S0027	1D	Gebouw D
S0028	1C	Gebouw Z

## 7.2 SELECT FROM WHERE

In Excel konden we geselecteerde kolommen zichtbaar maken door de niet-geselecteerde kolommen te verbergen. In SQL selecteren we juist de kolommen die we wel zichtbaar willen maken en blijven de niet geselecteerde kolommen verborgen.

Belangrijk is te realiseren dat we slechts de geselecteerde gegevens zichtbaar hebben gemaakt maar dat in de tabel zelf nog steeds alle data volledig aanwezig is. We zien dus (een deel van) een kopie van de gegevens uit de tabel en wijzigen niets in de tabel zelf.

Door WHERE toe te voegen aan het SELECT statement kunnen we selecteren welke rijen we zichtbaar willen maken, net zoals we in Excel de gewenste rijen kunnen filteren.

**SELECT <kolom1>, <kolom2>, ... FROM <tabel> WHERE ...;**

Achter WHERE kunnen we op verschillende manieren rijen selecteren, bv:

**SELECT \* FROM StudentenInKlassen WHERE Klas = '1A';**

Studentnummer	Voorletters	Achternaam	Geslacht	Geboortedatum	Adres	Huisnummer	Postcode	Woonplaats	Telefoonnummer	K.	Locatie	Adres	Postcode	Plaats	Klassedocent
S0001	A.	Baan	M	12-5-2004	Terwestenstraat	18	2525GH	Den Haag	06-12345678	1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P. Jansen
S0009	S.	Huber	M	12-5-2004	Hobbeistraat	22	2526JP	Den Haag	06-90123456	1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P. Jansen
S0013	A.D.	Eifering	M	5-9-2005	Reitstraat	52	2571RT	Den Haag	06-57913579	1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P. Jansen
S0018	D.	Pol	M	12-12-2004	Wesselsstraat	283	2572RZ	Den Haag		1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P. Jansen
S0019	E.	Reitsma	M	12-5-2004	Retiefstraat	27	2571PS	Den Haag	06-47890123	1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P. Jansen
S0022	L.	Joemman	M	12-5-2005	Hobbeistraat	20	2526JP	Den Haag	06-94567890	1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P. Jansen
S0025	O.	Snellers	M	12-10-2005	Herman Costerstraat	160	2571PD	Den Haag	06-30986543	1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P. Jansen

- = betekent gelijk aan
- <> of != betekent beiden ongelijk aan
- < betekent kleiner dan
- > betekent groter dan
- <= betekent kleiner of gelijk aan
- >= betekent groter of gelijk aan
- Bij selecteren op basis van letters (en cijfers als letters) moet het woord worden gezet tussen de ' ' tekens.
- Bij selecteren van cijfers (getallen met een waarde) moet er géén '' tekens worden gebruikt.

Andere voorbeelden:

**SELECT \* FROM StudentenInKlassen WHERE Woonplaats = 'Zoetermeer';**

Studentnummer	Voorletters	Achternaam	Geslach	geboortedatum	Adres	Huisnummer	Postcode	Woonplaats	telefoonnummer	K.	Locatie	Adres	Postcode	Plaats	Klassedoce
S0003	J.	Jansen	M	14-3-2004	Brinkershof	17	2713TX	Zoetermeer	071-1234567	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
S0004	J.	Jansen	M	14-3-2004	Brinkershof	17	2713TX	Zoetermeer	071-1234567	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen

**SELECT \* FROM StudentenInKlassen WHERE Studentnummer <= 'S005';**

Studentnummer	Voorletters	Achternaam	Geslach	geboortedatum	Adres	Huisnummer	Postcode	Woonplaats	telefoonnummer	K.	Locatie	Adres	Postcode	Plaats	Klassedoce
S0001	A.	Baan	M	12-5-2004	Terwestenstraat	18	2525GH	Den Haag	06-12345678	1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P. Jansen
S0002	S.	Hijmans	V	7-12-2004	Swelinkplein	16	2517GM	Den Haag	06-23456789	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0003	J.	Jansen	M	14-3-2004	Brinkershof	17	2713TX	Zoetermeer	071-1234567	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
S0004	J.	Jansen	M	14-3-2004	Brinkershof	17	2713TX	Zoetermeer	071-1234567	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
S0005	A.	Maarsen	M	12-5-2004	Rembrandtstraat	2	2526PZ	Den Haag	06-56789012	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten

Bij selecteren op basis van letters en cijfers gecombineerd wordt er alfanumeriek geselecteerd, waarbij cijfers als letters worden behandeld. Alfanumeriek sorteren betekent dat er gesorteerd word op volgorde 0123456789abcdefghijklmnpqrstuvwxyz.

### 7.3 SELECT FROM WHERE AND/OR

Met AND en OR kunnen we meerdere voorwaarden met elkaar combineren

**SELECT \* FROM StudentenInKlassen WHERE Geslacht = 'V' AND Klas = '1D';**

Studentnummer	Voorletters	Achternaam	Geslach	geboortedatum	Adres	Huisnummer	Postcode	Woonplaats	telefoonnummer	K.	Locatie	Adres	Postcode	Plaats	Klassedoce
S0021	K.	Hazenberg	V	12-5-2004	van der Vennestraat	83	2525CC	Den Haag	06-71234567	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
	H.	Blokhus	V	8-10-2006	Rembrandtstraat	3	2526PN	Den Haag	06-41234098	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten

**SELECT \* FROM StudentenInKlassen WHERE Geslacht = 'V' OR Klas = '1D';**

Studentnummer	Voorletters	Achternaam	Geslach	geboortedatum	Adres	Huisnummer	Postcode	Woonplaats	telefoonnummer	Klas	Locatie	Adres	Postcode	Plaats	Klassedoent
S0002	T.	Boeren	V	12-8-2004	Herman Costerstraat	164	2713PD	Den Haag	06-13456789	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0005	D.	Haverkamp	V	12-5-2004	Retiefstraat	25	2571PS	Den Haag	06-52345679	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0006	S.	Hijmans	V	7-12-2004	Swelinkplein	16	2517GM	Den Haag	06-23456789	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0010	K.	Jakobs	V	12-12-2005	De Vliegerstraat	55	2525BG	Den Haag	06-67890123	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0011	Z.	Wessels	V	5-8-2004	Retiefstraat	29	2713PS	Den Haag	06-13579135	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0015	H.	Blokhus	V	8-10-2006	Rembrandtstraat	3	2526PN	Den Haag	06-41234098	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
S0016	B.	Bol	M	12-12-2006	AAAAAAAAAAAA	22	2525GH	Den Haag	06-59873456	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
S0017	J.	de Jong	M	12-12-2006	van Miereveldstraat	62	2525EG	Den Haag	06-91357913	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
S0021	W.	de Wit	M	12-5-2004	Almeloplein	10	2533AA	Den Haag	06-29876543	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
S0024	K.	Hazenberg	V	12-5-2004	van der Vennestraat	83	2525CC	Den Haag	06-1234567	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
S0026	J.	Huizinga	M	12-5-2005	Terwestenstraat	20	2525GH	Den Haag	06-25678901	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
S0026	A.	Kassenberg	M	12-5-2006	Almeloplein	12	2533AA	Den Haag	06-24680246	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
S0027	A.	Maarsen	M	12-5-2004	Rembrandtstraat	2	2526PZ	Den Haag	06-56789012	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
	K.	Reeuvenkamp	M	22-11-2005	Reitzstraat	50	2571RT	Den Haag	06-59873112	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten

Met **AND** geven we aan dat er aan beide voorwaarden moet worden voldaan.

Met **OR** geven we aan dat er aan één of beide van de voorwaarden moet worden voldaan

Let bij het gebruik van AND en OR op dat AND voor gaat op OR, net zoals bij het rekenen vermenigvuldigen voor gaat op optellen:

$$1 + 2 * 3 = 1 + (2 * 3) = 1 + 6 = 7$$

En niet:

$$1 + 2 * 3 = (1 + 2) * 3 = 3 * 3 = 9$$

Zo geldt ook:

voorwaarde1 OR voorwaarde2 AND voorwaarde3 = voorwaarde1 OR (voorwaarde2 AND voorwaarde3)

En niet:

voorwaarde1 OR voorwaarde2 AND voorwaarde3 = (voorwaarde1 OR voorwaarde2) AND voorwaarde3

Indien je wilt dat de query anders met deze regels omgaat, moeten er net zoals bij het rekenen haken () worden gebruikt.

Voorbeeld:

**SELECT \* FROM StudentenInKlassen WHERE (Geslacht = 'V' OR Geslacht = 'M') AND Klas = '1B';**

Studentnummer	Voorletter	Achternaam	Geslach	geboortedatum	Adres	Huisnummer	Postcode	Woonplaats	telefoonnummer	K.	Locatie	Adres	Postcode	Plaats	Klassedoce
S0002	S.	Hijmans	V	7-12-2004	Sweelinckplein	16	2517GM	Den Haag	06-23456789	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0006	K.	Jakobs	V	12-12-2005	De Vliegerstraat	55	2525BG	Den Haag	06-67890123	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0007	A.B.	Willemsen	M	14-5-2004	Terwestenstraat	16	2525GH	Den Haag	06-78901234	1B	Gebouw H	Wegastraat 56	2516CJ	Den Haag	A.P. Jansen
S0010	Z.	Wessels	V	5-8-2004	Retiefstraat	29	2571PS	Den Haag	06-13579135	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0016	T.	Boeren	V	12-8-2004	Herman Costerstraat	164	2571PD	Den Haag	06-13456789	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0020	P.	Vos	M	5-3-2004	van Miereveldstraat	64	2525EG	Den Haag	06-69012345	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0026	D.	Haverkamp	V	12-5-2004	Retiefstraat	25	2571PS	Den Haag	06-52345679	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer

Let dus op dat het weglaten van de haken een ander resultaat geeft:

**SELECT \* FROM StudentenInKlassen WHERE Geslacht = 'V' OR Geslacht = 'M' AND Klas = '1B';**

Studentnummer	Voorletter	Achternaam	Geslach	geboortedatum	Adres	Huisnummer	Postcode	Woonplaats	telefoonnummer	K.	Locatie	Adres	Postcode	Plaats	Klassedoce
S0002	S.	Hijmans	V	7-12-2004	Sweelinckplein	16	2517GM	Den Haag	06-23456789	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0003	J.	Jansen	M	14-3-2004	Brinkershof	17	2713TM	Zoetermeer	071-1234567	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
S0004	J.	Jansen	M	14-3-2004	Brinkershof	17	2713TM	Zoetermeer	071-1234567	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
S0006	K.	Jakobs	V	12-12-2005	De Vliegerstraat	55	2525BG	Den Haag	06-67890123	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0008	U.	Jans	M	17-6-2005	Herman Costerstraat	162	2571PD	Den Haag	06-89012345	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
S0010	Z.	Wessels	V	5-8-2004	Retiefstraat	29	2571PS	Den Haag	06-13579135	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0012	L.	Vrogers	M	12-5-2004	van der Vennestraat	85	2525CC	Den Haag	06-35791357	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
S0014	I.	Sebastiaans	M	12-5-2004	Rembrandtstraat	1	2526PN	Den Haag	06-79135791	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
S0016	T.	Boeren	V	12-8-2004	Herman Costerstraat	164	2571PD	Den Haag	06-13456789	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0021	K.	Hazenberg	V	12-5-2004	van der Vennestraat	83	2525CC	Den Haag	06-71234567	1D	Gebouw D	Oude Delft 12	2611OC	Delft	S. Slooten
S0026	D.	Blokhus	V	8-10-2006	Rembrandtstraat	3	2526PN	Den Haag	06-41234098	1D	Gebouw D	Oude Delft 12	2611OC	Delft	S. Slooten
S0028	L.	Haverkamp	V	12-5-2004	Retiefstraat	25	2571PS	Den Haag	06-52345679	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
		van den Broek	M	12-5-2004	Wesselstraat	281	2572RZ	Den Haag	06-22095586	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen

In het eerste voorbeeld worden alle mannen en vrouwen uit klas 1B geselecteerd.

In het tweede voorbeeld worden de mannen uit klas 1B geselecteerd, maar ook alle vrouwen ongeacht de klas waar ze in zitten!

#### 7.4 SELECT FROM WHERE AND/OR ORDER BY

Met ORDER BY kunnen we het resultaat van de query sorteren op een kolom of op een combinatie van kolommen:

**SELECT \* FROM StudentenInKlassen WHERE Geslacht = 'V' ORDER BY Achternaam;**

Studentnummer	Voorletters	Achternaam	Geslach	geboortedatum	Adres	Huisnummer	Postcode	Woonplaats	telefoonnummer	K.	Locatie	Adres	Postcode	Plaats	Klassedoce
S0002	S.	Blokhus	V	7-12-2004	Sweelinckplein	16	2517GM	Den Haag	06-23456789	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0006	K.	Boeren	V	12-12-2005	De Vliegerstraat	55	2525BG	Den Haag	06-67890123	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0010	Z.	Haverkamp	V	5-8-2004	Retiefstraat	29	2571PS	Den Haag	06-13579135	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0016	T.	Hazenberg	V	12-8-2004	Herman Costerstraat	164	2571PD	Den Haag	06-13456789	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0021	K.	Hijmans	V	12-5-2004	van der Vennestraat	83	2525CC	Den Haag	06-71234567	1D	Gebouw D	Oude Delft 12	2611OC	Delft	S. Slooten
S0026	D.	Jakobs	V	8-10-2006	Rembrandtstraat	3	2526PN	Den Haag	06-41234098	1D	Gebouw D	Oude Delft 12	2611OC	Delft	S. Slooten
		Wessels	V	12-5-2004	Retiefstraat	25	2571PS	Den Haag	06-52345679	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer

We kunnen het resultaat ook over meerdere kolommen sorteren, bijvoorbeeld eerst sorteren op klas en dan binnen die sortering sorteren op Achternaam:

**SELECT \* FROM StudentenInKlassen ORDER BY Klas, Achternaam;**

Studentnummer	Voorletters	Achternaam	Geslacht	geboortedatum	Adres	Huisnummer	Postcode	Woonplaats	telefoonnummer	Klas	Locatie	Adres	Postcode	Plaats	Klassedocent
S0001	A.	Baan	M	12-5-2004	Terwestenstraat	18	2525GH	Den Haag	06-12345678	1A	Gebouw H	Wegastraat 56	2516CJ	Den Haag	A.P. Jansen
S0013	A.D.	Eifering	M	5-9-2005	Reitzstraat	52	2571RT	Den Haag	06-57913579	1A	Gebouw H	Wegastraat 56	2516CJ	Den Haag	A.P. Jansen
S0009	S.	Huber	M	12-5-2004	Hobbemastraat	22	2526JP	Den Haag	06-90123456	1A	Gebouw H	Wegastraat 56	2516CJ	Den Haag	A.P. Jansen
S0022	L.	Joemman	M	12-5-2005	Hobbemastraat	20	2526JP	Den Haag	06-94567890	1A	Gebouw H	Wegastraat 56	2516CJ	Den Haag	A.P. Jansen
S0018	D.	Pol	M	12-12-2004	Wesselsstraat	283	2572RZ	Den Haag		1A	Gebouw H	Wegastraat 56	2516CJ	Den Haag	A.P. Jansen
S0019	E.	Reitsema	M	12-5-2004	Retiefstraat	27	2571PS	Den Haag	06-47890123	1A	Gebouw H	Wegastraat 56	2516CJ	Den Haag	A.P. Jansen
S0025	O.	Snellers	M	12-10-2005	Herman Costerstraat	160	2571PD	Den Haag	06-30986543	1A	Gebouw H	Wegastraat 56	2516CJ	Den Haag	A.P. Jansen
S0016	T.	Boeren	V	12-8-2004	Herman Costerstraat	164	2571PD	Den Haag	06-13456789	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0026	D.	Haverkamp	V	12-5-2004	Retiefstraat	25	2571PS	Den Haag	06-52345679	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0002	S.	Hijmans	V	7-12-2004	Sweelinckplein	16	2517GM	Den Haag	06-23456789	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0006	K.	Jakobs	V	12-12-2005	De Vliegerstraat	55	2525BG	Den Haag	06-67890123	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0020	P.	Vos	M	5-3-2004	Van Miereveldstraat	64	2525EG	Den Haag	06-69012345	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0010	Z.	Wessels	V	5-8-2004	Retiefstraat	29	2571PS	Den Haag	06-13579135	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0007	A.B.	Willemsen	M	14-5-2004	Terwestenstraat	16	2525GH	Den Haag	06-78901234	1B	Gebouw H	Wegastraat 56	2516CJ	Den Haag	A.P. Jansen
S0023	S.	Claasen	-	8-9-2004	Terwestenstraat	111	2525GG	Den Haag	06-10987654	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
S0008	U.	Jans	M	17-6-2005	Herman Costerstraat	162	2571PD	Den Haag	06-89012345	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
S0003	J.	Jansen	M	14-3-2004	Brinkershof	17	2713TX	Zoetermeer	071-1234567	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
S0004	J.	Jansen	M	14-3-2004	Brinkershof	17	2713TX	Zoetermeer	071-1234567	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
S0014	I.	Sebastiaans	M	12-5-2004	Rembrandtstraat	1	2526PN	Den Haag	06-79135791	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
S0028	L.	van den Broek	M	12-5-2004	Wesselsstraat	281	2572RZ	Den Haag	06-22095586	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
S0012	L.	Vrogers	M	12-5-2004	van der Vennestraat	85	2525CC	Den Haag	06-35791357	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
S0026	H.	Blokhuis	V	8-10-2006	Rembrandtstraat	3	2526PN	Den Haag	06-41234098	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
S0015	B.	Bol	M	12-12-2006	AAAAAAAAAAAAAA	22	2525GH	Den Haag	06-59873456	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
S0001	J.	de Jong	M	12-12-2006	van Miereveldstraat	62	2525EG	Den Haag	06-91357913	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
S0024	W.	de Wit	M	12-5-2004	Almeloplein	10	2533AA	Den Haag	06-29876543	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
S0021	K.	Hasenberg	V	12-5-2004	van der Vennestraat	83	2525CC	Den Haag	06-71234567	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
S0017	J.	Huizinga	M	12-5-2005	Terwestenstraat	20	2525GH	Den Haag	06-25678901	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
S0011	A.	Kassenberg	M	12-5-2006	Almeloplein	12	2533AA	Den Haag	06-24680246	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
S0005	A.	Maarsen	M	12-5-2004	Rembrandtstraat	2	2526PZ	Den Haag	06-56789012	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
S0027	K.	Reuvenkamp	M	22-11-2005	Reitzstraat	50	2571RT	Den Haag	06-59873112	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten

Met AS kunnen we de namen van de kolommen in het rapport veranderen.

**SELECT Voorletters, Achternaam AS Familienaam FROM StudentenInKlas;**

Voorletters	Familienaam
A.	Baan
S.	Hijmans
J.	Jansen
J.	Jansen
A.	Maarsen
K.	Jakobs
A.B.	Willemsen
U.	Jans
S.	Huber
Z.	Wessels
A.	Kassenberg
L.	Vrogers
A.D.	Eifering
I.	Sebastiaans
J.	de Jong
T.	Boeren
J.	Huizinga
D.	Pol
E.	Reitsema
P.	Vos
K.	Hasenberg
L.	Joemman
S.	Claasen
W.	de Wit
O.	Snellers
H.	Blokhuis
D.	Haverkamp
B.	Bol
K.	Reuvenkamp
L.	van den Broek

Ook hierbij is het belangrijk te realiseren dat we de naam van de kolom in het rapport hebben veranderd, maar de kolomnaam in de tabel zelf ongewijzigd blijft.

## 7.5 SELECT DISTINCT FROM WHERE AND/OR ORDER BY

Met DISTINCT kunnen we zo nodig dubbele gegevens weglaten

**SELECT Geslacht FROM StudentenInKlassen;**

Geslacht
M
V
M
M
M
V
M
M
M
M
V
M
M
M
M
V
M
M
M
M
V
M
-
M
M
V
V
M
M

**SELECT DISTINCT Geslacht FROM StudentenInKlassen;**

Geslacht
M
V
-

## 7.6 SELECT FROM WHERE NOT AND/OR ORDER BY

Met **NOT** kunnen we het tegengestelde resultaat verkrijgen van wat achter de NOT staat:

**SELECT \* FROM StudentenInKlas WHERE NOT Geslacht = 'M';**

Dit geeft alle rijen waarbij geldt dat het geslacht niet mannelijke is.

Studentnummer	Voorletters	Achternaam	Geslacht	Geboortedatum	Adres	Huisnummer	Postcode	Woonplaats	telefoonnummer	Klas	Locatie	Adres	Postcode	Plaats	Klassedocent
S0002	S.	Hijmans	V	7-12-2004	Sweelinckplein	16	2517GM	Den Haag	06-23456789	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0006	K.	Jakobs	V	12-12-2005	De Vliegerstraat	55	2525BG	Den Haag	06-67890123	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0010	Z.	Wessels	V	5-8-2004	Retiefstraat	29	2571PS	Den Haag	06-13579135	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0016	T.	Boeren	V	12-8-2004	Herman Costerstraat	164	2571PD	Den Haag	06-13456789	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0021	K.	Hasenberg	V	12-5-2004	van der Vennestraat	83	2525CC	Den Haag	06-71234567	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
S0023	S.	Claasen	-	8-9-2004	Terwestenstraat	111	2525GG	Den Haag	06-10987654	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
H.	Blokhuis	V	V	8-10-2006	Rembrandtstraat	3	2526PN	Den Haag	06-41234098	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
S0026	D.	Haverkamp	V	12-5-2004	Retiefstraat	25	2571PS	Den Haag	06-52345679	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer

## 7.7 NULL waarde

In databases kennen we een waarde **NULL**. Deze waarde is niet gelijk aan het getal 0 en is ook niet gelijk aan het woord '**NULL**'. De waarde **NULL** betekent dat het veld leeg ofwel niet ingevuld is. De waarde daarvan is daardoor onbekend!

Selecteren op kolommen met **NULL** waarden kan daardoor soms een onverwacht resultaat geven.

Stel we hebben een tabel met de naam **Namen**:

**SELECT \* FROM Namen;**

Voornaam	Achternaam
Jan	Jansen
Klaas	NULL
Pietje	Bel

In de rij met de voornaam Klaas staat voor de kolom Achternaam dus niets ingevuld. De waarde in dat veld is dus **NULL** (leeg, ofwel onbekend).

**SELECT \* FROM Namen WHERE Achternaam IS NULL;**

### Voornaam Achternaam

Klaas	NULL
-------	------

Wanneer we explicet zoeken naar de waarde NULL, dan komt deze rij in het resultaat.

`SELECT * FROM Namen WHERE Achternaam = 'Jansen';`

### Voornaam Achternaam

Jan	Jansen
-----	--------

`SELECT * FROM Namen WHERE Achternaam <> 'Jansen';`

### Voornaam Achternaam

Pietje	Bel
--------	-----

Deze laatste twee queries geven een onverwacht resultaat. De achternaam van Kees is dus niet gelijk aan 'Jansen' maar ook niet ongelijk aan 'Jansen'! Dit komt omdat de achternaam de waarde NULL ofwel onbekend heeft. NULL is dus een derde mogelijkheid naast gelijk of ongelijk.

Twee voorwaarden is dus niet voldoende:

- Is het gelijk aan ...?
- Is het ongelijk aan ...?

Maar er moet nog een derde voorwaarde bij worden gesteld:

- Is de waarde NULL?

Een juiste query zou dan zijn:

`SELECT * FROM Namen WHERE Achternaam <> 'Jansen' OR Achternaam IS NULL;`

### Voornaam Achternaam

Klaas	NULL
Pietje	Bel

Houdt daarom altijd rekening met velden met NULL waarden! De query kan anders onverwachte (onjuiste) resultaten geven.

## 7.8 IS NULL en IS NOT NULL

Met IS NULL kunnen we alle rijen selecteren waarvan de waarde in een veld NULL is.

`SELECT * FROM StudentenInKlassen where Studentnummer IS NULL;`

Dit geeft alle rijen waarvan de kolom Studentnummer NULL (niet ingevuld) is.

Studentnummer	Voorletters	Achternaam	Geslach	geboortedatum	Adres	Huisnummer	Postcode	Woonplaats	telefoonnummer	Klas	Locatie	Adres	Postcode	Plaats	Klassedocent
H.	Blokhuis	V		8-10-2006	Rembrandtstraat	3	2526PN	Den Haag	06-41234098	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten

`SELECT * FROM StudentenInKlassen where Studentnummer IS NOT NULL;` en

`SELECT * FROM StudentenInKlassen where NOT Studentnummer IS NULL;`

Deze twee queries geven beiden hetzelfde resultaat waarvan de kolom Studentnummer niet de waarde NULL heeft.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
Studentnummer	Voorletters	Achternaam	Geslacht	geboortedatum	Adres	Huisnummer	Postcode	Woonplaats	telefoonnummer	Klas	Locatie	Adres	Postcode	Plaats	Klassedocent
S0001	A.	Baan	M	12-5-2004	Terwestenstraat	18	2525GH	Den Haag	06-12345678	1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P. Jansen
S0002	S.	Hijmans	V	7-12-2004	Sweelinckplein	16	2517GM	Den Haag	06-23456789	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0003	J.	Jansen	M	14-3-2004	Brinkershof	17	2713TX	Zoetermeer	071-1234567	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
S0004	J.	Jansen	M	14-3-2004	Brinkershof	17	2713TX	Zoetermeer	071-1234567	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
S0005	A.	Maarsen	M	12-5-2004	Rembrandtstraat	2	2526PZ	Den Haag	06-56789012	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
S0006	K.	Jakobs	V	12-12-2005	De Vliegerstraat	55	2525BG	Den Haag	06-67890123	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0007	A.B.	Willemsen	M	14-5-2004	Terwestenstraat	16	2525GH	Den Haag	06-78901234	1C	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P. Jansen
S0008	U.	Jans	M	17-6-2005	Herman Costerstraat	162	2571PD	Den Haag	06-89012345	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
S0009	S.	Huber	M	12-5-2004	Hobbeistraat	22	2526JP	Den Haag	06-90123456	1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P. Jansen
S0010	Z.	Wessels	V	5-8-2004	Retiefstraat	29	2571PS	Den Haag	06-13579135	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0011	A.	Kassenberg	M	12-5-2006	Almeloplein	12	2533AA	Den Haag	06-24680246	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
S0012	L.	Vrogers	M	12-5-2004	van der Vennestraat	85	2525CC	Den Haag	06-35791357	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
S0013	A.D.	Elfering	M	5-9-2005	Retiefstraat	52	2571RT	Den Haag	06-57913579	1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P. Jansen
S0014	I.	Sebastiaans	M	12-5-2004	Rembrandtstraat	1	2526PN	Den Haag	06-79135791	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
S0015	J.	de Jong	M	12-12-2006	van Miereveldstraat	62	2525EG	Den Haag	06-91357913	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
S0016	T.	Boeren	V	12-8-2004	Herman Costerstraat	164	2571PD	Den Haag	06-13456789	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0017	J.	Huijzinga	M	12-5-2005	Terwestenstraat	20	2525GH	Den Haag	06-25678901	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
S0018	D.	Pol	M	12-12-2004	Wesselstraat	283	2572RZ	Den Haag		1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P. Jansen
S0019	E.	Reitsema	M	12-5-2004	Retiefstraat	27	2571PS	Den Haag	06-47890123	1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P. Jansen
S0020	P.	Vos	M	5-3-2004	van Miereveldstraat	64	2525EG	Den Haag	06-69012345	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0021	K.	Hasenberg	V	12-5-2004	van der Vennestraat	83	2525CC	Den Haag	06-71234567	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
S0022	L.	Joemman	M	12-5-2005	Hobbeistraat	20	2526JP	Den Haag	06-94567890	1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P. Jansen
S0023	S.	Claasen	-	8-9-2004	Terwestenstraat	111	2525GG	Den Haag	06-10987654	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
S0024	W.	de Wit	M	12-5-2004	Almeloplein	10	2533AA	Den Haag	06-29876543	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
S0025	O.	Snellers	M	12-10-2005	Herman Costerstraat	160	2571PD	Den Haag	06-30986543	1A	Gebouw H	Wegstraat 56	2516CJ	Den Haag	A.P. Jansen
S0026	D.	Haverkamp	V	12-5-2004	Retiefstraat	25	2571PS	Den Haag	06-52345679	1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
S0026	B.	Bol	M	12-12-2006	AAAAAAAAAAAAAA	22	2525GH	Den Haag	06-59873456	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
S0027	K.	Reuvenkamp	M	22-11-2005	Retiefstraat	50	2571RT	Den Haag	06-59873112	1D	Gebouw D	Oude Delft 12	2611CC	Delft	S. Slooten
S0028	L.	van den Broek	M	12-5-2004	Wesselstraat	281	2572RZ	Den Haag	06-22095586	1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen

Let altijd op dat je niet een = teken gebruikt in plaats van het woord IS!

```
SELECT * FROM Namens WHERE Achternaam = NULL;
SELECT * FROM Namens WHERE Achternaam = 'NULL';
```

Deze twee queries zullen géén resultaat geven daar het woord NULL niet in de kolom ingevuld is.

```
SELECT * FROM Namens WHERE Achternaam IS NULL;
```

Voornaam	Achternaam
Klaas	NULL

Deze derde query geeft wel een resultaat waar de waarde NULL (leeg) is.

## 7.9 SELECT INTO (NEW)

Het komt wel eens voor dat we de inhoud van een tabel naar een nieuwe tabel willen kopiëren. Dat kan met het SELECT INTO commando.

Voorbeeld:

```
USE Oefeningen;
CREATE TABLE Tabel13 (
een INT,
twee VARCHAR(20)
);
INSERT INTO Tabel13 VALUES (1, 'eerste');
INSERT INTO Tabel13 VALUES (2, 'tweede');
INSERT INTO Tabel13 VALUES (3, 'derde');
SELECT een, twee INTO Tabel14 FROM Tabel13 WHERE een > 2;
SELECT * FROM Tabel13;
SELECT * FROM Tabel14;
INSERT INTO TABEL14 VALUES(SELECT * FROM Tabel13);
```

SQLQuery26.sql - M...(USR\NIJP02 (52))\*

```
een INT,  
twee VARCHAR(20)  
);  
INSERT INTO Tabel13 VALUES (1,'eerste');  
INSERT INTO Tabel13 VALUES (2,'tweede');  
INSERT INTO Tabel13 VALUES (3,'derde');  
SELECT een, twee INTO Tabel14 FROM Tabel13 WHERE een > 2;  
SELECT * FROM Tabel13;  
SELECT * FROM Tabel14;  
INSERT INTO TABEL14 VALUES(SELECT * FROM Tabel13);
```

179 %

	een	twee
1	1	eerste
2	2	tweede
3	3	derde

## Hoofdstuk 8 Aggregatie functies

### 8.1 Aggregatie functies

Met aggregatie functies kunnen we resultaten samenvoegen tot een groter geheel.

De aggregatie functies die wij in dit hoofdstuk zullen behandelen zijn:

- COUNT = telt het aantal rijen
- SUM = berekent de optelsom van de waarde in de kolom
- AVG = berekent de gemiddelde waarde van de waarden in de kolom
- MAX = berekent de maximale waarde in de kolom
- MIN = berekent de minimale waarde in de kolom
- IN = zoekt naar een waarden binnen een opgegeven reeks tussen haakjes ()
- BETWEEN = zoekt naar waarden binnen een aaneengesloten reeks
- LIKE = zoekt naar een string letters die in het veld voorkomen

Als voorbeeld nemen we de tabel **Products** in de database **Nortwind**, met de kolommen:

ProductID	= Uniek ID van elk product
ProductName	= Naam van het product
Supplier	= Leverancier
CategoryID	= Category waarin product valt
QuantityPer Unit	= Aantal stuks per eenheid
UnitPrice	= Prijs per eenheid
UnitsInStock	= Eenheden op voorraad in het magazijn
UnitsInOrder	= Eenheden die in bestelling zijn
ReorderLevel	= Voorraadniveau waarbij nieuwe bestelling moet worden geplaatst
Discontinued	= Stopgezet

In de tabel zitten 77 rijen en 10 kolommen:

**SELECT \* FROM Products;**

ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel	Discontinued
1	Chai	1	1	10 boxes x 20 bags	18,00	39	0	10	0
2	Chang	1	1	24 - 12 oz bottles	19,00	17	40	25	0
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10,00	13	70	25	0
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22,00	53	0	0	0
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21,35	0	0	0	1
6	Grandma's Boysenberry Spread	3	2	12 - 8 oz jars	25,00	120	0	25	0
7	Uncle Bob's Organic Dried Pears	3	7	12 - 1 lb pkgs.	30,00	15	0	10	0
8	Northwoods Cranberry Sauce	3	2	12 - 12 oz jars	40,00	6	0	0	0
9	Mishi Kobe Niku	4	6	18 - 500 g pkgs.	97,00	29	0	0	1
10	Ikura	4	8	12 - 200 ml jars	31,00	31	0	0	0
11	Queso Cabrales	5	4	1 kg pkg.	21,00	22	30	30	0
12	Queso Manchego La Pastora	5	4	10 - 500 g pkgs.	38,00	86	0	0	0
13	Konbu	6	8	2 kg box	6,00	24	0	5	0
14	Tofu	6	7	40 - 100 g pkgs.	23,25	35	0	0	0
15	Genen Shouyu	6	2	24 - 250 ml bottles	15,50	39	0	5	0
16	Pavlova	7	3	32 - 500 g boxes	17,45	29	0	10	0
17	Alice Mutton	7	6	20 - 1 kg tins	39,00	0	0	0	1
18	Carnarvon Tigers	7	8	16 kg pkg.	62,50	42	0	0	0
19	Teatime Chocolate Biscuits	8	3	10 boxes x 12 pieces	9,20	25	0	5	0
20	Sir Rodney's Marmalade	8	3	30 gift boxes	81,00	40	0	0	0
21	Sir Rodney's Scones	8	3	24 pkgs. x 4 pieces	10,00	3	40	5	0
22	Gustaf's Knäckebrot	9	5	24 - 500 g pkgs.	21,00	104	0	25	0
23	Tunnbröd	9	5	12 - 250 g pkgs.	9,00	61	0	25	0
24	Guaraná Fantástica	10	1	12 - 355 ml cans	4,50	20	0	0	1
25	NuNuCa Nuß-Nougat-Creme	11	3	20 - 450 g glasses	14,00	76	0	30	0
26	Gumbär Gummibärchen	11	3	100 - 250 g bags	31,23	15	0	0	0
27	Schoggi Schokolade	11	3	100 - 100 g pieces	43,90	49	0	30	0
28	Rössle Sauerkraut	12	7	25 - 825 g cans	45,60	26	0	0	1
29	Thüringer Rostbratwurst	12	6	50 bags x 30 sausgs.	123,79	0	0	0	1
30	Nord-Ost Matjeshering	13	8	10 - 200 g glasses	25,89	10	0	15	0
31	Gorgonzola Telino	14	4	12 - 100 g pkgs.	12,50	0	70	20	0
32	Mascarpone Fabioli	14	4	24 - 200 g pkgs.	32,00	9	40	25	0
33	Geitost	15	4	500 g	2,50	112	0	20	0
34	Sasquatch Ale	16	1	24 - 12 oz bottles	14,00	111	0	15	0
35	Steeleye Stout	16	1	24 - 12 oz bottles	18,00	20	0	15	0
36	Inlagd Sill	17	8	24 - 250 g jars	19,00	112	0	20	0
37	Gravad lax	17	8	12 - 500 g pkgs.	26,00	11	50	25	0
38	Côte de Blaye	18	1	12 - 75 cl bottles	263,50	17	0	15	0
39	Chartreuse verte	18	1	750 cc per bottle	18,00	69	0	5	0
40	Boston Crab Meat	19	8	24 - 4 oz tins	18,40	123	0	30	0
41	Jack's New England Clam Chowder	19	8	12 - 12 oz cans	9,65	85	0	10	0
42	Singaporen Hokkien Fried Mee	20	5	32 - 1 kg pkgs.	14,00	26	0	0	1
43	Ipo Coffee	20	1	16 - 500 g tins	46,00	17	10	25	0
44	Gula Malacca	20	2	20 - 2 kg bags	19,45	27	0	15	0
45	Røgede sild	21	8	1k pkg.	9,50	5	70	15	0
46	Spegesild	21	8	4 - 450 g glasses	12,00	95	0	0	0
47	Zaanse koeken	22	3	10 - 4 oz boxes	9,50	36	0	0	0
48	Chocolade	22	3	10 pkgs.	12,75	15	70	25	0
49	Maxikaku	23	3	24 - 50 g pkgs.	20,00	10	60	15	0
50	Valkoinen suklaa	23	3	12 - 100 g bars	16,25	65	0	30	0
51	Manjimup Dried Apples	24	7	50 - 300 g pkgs.	53,00	20	0	10	0
52	Filo Mix	24	5	16 - 2 kg boxes	7,00	38	0	25	0
53	Pent Pasties	24	6	48 pieces	32,80	0	0	0	1
54	Tourtière	25	6	16 pies	7,45	21	0	10	0
55	Pâté chinois	25	6	24 boxes x 2 pies	24,00	115	0	20	0
56	Gnocchi di nonna Alice	26	5	24 - 250 g pkgs.	38,00	21	10	30	0
57	Ravioli Angelo	26	5	24 - 250 g pkgs.	19,50	36	0	20	0
58	Escargots de Bourgogne	27	8	24 pieces	13,25	62	0	20	0
59	Radlette Courdavault	28	4	5 kg pkg.	55,00	79	0	0	0
60	Camembert Pierrot	28	4	15 - 300 g rounds	34,00	19	0	0	0
61	Sirop d'éráble	29	2	24 - 500 ml bottles	28,50	113	0	25	0
62	Tarte au sucre	29	3	48 pies	49,30	17	0	0	0
63	Vegie-spread	7	2	15 - 625 g jars	43,90	24	0	5	0
64	Wimmer's gute Semmelknödel	12	5	20 bags x 4 pieces	33,25	22	80	30	0
65	Louisiana Fiery Hot Pepper Sauce	2	2	32 - 8 oz bottles	21,05	76	0	0	0
66	Louisiana Hot Spiced Okra	2	2	24 - 8 oz jars	17,00	4	100	20	0
67	Laughing Lumberjack Lager	16	1	24 - 12 oz bottles	14,00	52	0	10	0
68	Scottish Longbreads	8	3	10 boxes x 8 pieces	12,50	6	10	15	0
69	Gudbrandsdalsost	15	4	10 kg pkg.	36,00	26	0	15	0
70	Outback Lager	7	1	24 - 355 ml bottles	15,00	15	10	30	0
71	Flotemysost	15	4	10 - 500 g pkgs.	21,50	26	0	0	0
72	Mozzarella di Giovanni	14	4	24 - 200 g pkgs.	34,80	14	0	0	0
73	Röd Kaviar	17	8	24 - 150 g jars	15,00	101	0	5	0
74	Longlife Tofu	4	7	5 kg pkg.	10,00	4	20	5	0
75	Rhönbräu Klosterbier	12	1	24 - 0.5 l bottles	7,75	125	0	25	0
76	Lakakkikööri	23	1	500 ml	18,00	57	0	20	0
77	Original Frankfurter grüne Soße	12	2	12 boxes	13,00	32	0	15	0

## 8.2 COUNT

Met COUNT kunnen we het aantal rijen laten tellen:

**SELECT COUNT(\*) FROM Products;**

Dit geeft als resultaat het getal 77 als totaal aantal producten (rijen) in de tabel Products.

	(No column name)
1	77

**SELECT COUNT(\*) FROM Products WHERE UnitsInStock = 0;**

Dit geeft als resultaat het getal 5 als totaal aantal producten waarvan de voorraad in het magazijn 0 zijn in de tabel Products.

	(No column name)
1	5

## 8.3 SUM

Met SUM kunnen we alle getallen van een kolom bij elkaar optellen:

**SELECT SUM(UnitsInStock) FROM Products;**

Dit geeft als resultaat het getal 3119 als totaal aantal eenheden in voorraad in het magazijn van alle producten samen.

	(No column name)
1	3119

**SELECT SUM(UnitsInStock) FROM Products where Discontinued = 1;**

Dit geeft als resultaat het getal 101 als totaal aantal eenheden in het magazijn waarvan het product stopgezet is.

	(No column name)
1	101

## 8.4 AVG

Met AVG kan de gemiddelde waarde van de getallen in een kolom worden berekend:

**SELECT AVG(UnitsInStock) FROM Products;**

Dit geeft als resultaat het getal 40 als gemiddeld aantal eenheden van alle producten in het magazijn.

	(No column name)
1	40

## 8.5 MAX

Met MAX kan het maximale getal in een kolom worden berekend:

**SELECT MAX(UnitsInStock) FROM Products;**

Dit geeft als resultaat het getal 125 als maximale voorraad van een product in het magazijn.

	(No column name)
1	125

## 8.6 MIN

Met MIN kan het minimale getal in een kolom worden berekend:

**SELECT MIN(UnitsInStock) FROM Products;**

Dit geeft als resultaat het getal 0 als minimale voorraad van een product in het magazijn.

	(No column name)
1	0

## 8.7 IN

Met IN kunnen we een reeks met waarden invullen waaraan de voorwaarde moet voldoen:

**SELECT \* FROM Products WHERE SupplierID IN (1,3,5);**

Dit geeft alle producten waarvan de leverancier als ID 1, 3 of 5 heeft.

	ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel	Discontinued
1	1	Chai	1	1	10 boxes x 20 bags	18.00	39	0	10	0
2	2	Chang	1	1	24 - 12 oz bottles	19.00	17	40	25	0
3	3	Aniseed Syrup	1	2	12 - 550 ml bottles	10.00	13	70	25	0
4	6	Grandma's Boysenberry Spread	3	2	12 - 8 oz jars	25.00	120	0	25	0
5	7	Uncle Bob's Organic Dried Pears	3	7	12 - 1 lb pkgs.	30.00	15	0	10	0
6	8	Northwoods Cranberry Sauce	3	2	12 - 12 oz jars	40.00	6	0	0	0
7	11	Queso Cabrales	5	4	1 kg pkg.	21.00	22	30	30	0
8	12	Queso Manchego La Pastora	5	4	10 - 500 g pkgs.	38.00	86	0	0	0

## 8.8 BETWEEN

Met BETWEEN kunnen we een aaneensluitende reeks met waarden tussen een minimale en een maximale waarde invullen waaraan de voorwaarde moet voldoen:

**SELECT \* FROM Products WHERE SupplierID BETWEEN 3 AND 5;**

Dit geeft alle rijen waarvan de leverancier een ID heeft tussen de 3 en de 5.

	ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel	Discontinued
1	6	Grandma's Boysenberry Spread	3	2	12 - 8 oz jars	25.00	120	0	25	0
2	7	Uncle Bob's Organic Dried Pears	3	7	12 - 1 lb pkgs.	30.00	15	0	10	0
3	8	Northwoods Cranberry Sauce	3	2	12 - 12 oz jars	40.00	6	0	0	0
4	9	Mishi Kobe Niku	4	6	18 - 500 g pkgs.	97.00	29	0	0	1
5	10	Ikura	4	8	12 - 200 ml jars	31.00	31	0	0	0
6	11	Queso Cabrales	5	4	1 kg pkg.	21.00	22	30	30	0
7	12	Queso Manchego La Pastora	5	4	10 - 500 g pkgs.	38.00	86	0	0	0
8	74	Longlife Tofu	4	7	5 kg pkg.	10.00	4	20	5	0

## 8.9 LIKE

Met LIKE kunnen we zoeken naar velden waarin een opgegeven string staat:

**SELECT \* FROM Employees WHERE Title LIKE '%SALES%';**

Dit toont alle medewerkers waarvan de functie het woord SALES bevat.

Het %-teken staat voor 0, 1 of meerdere karakters

EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	BirthDate	HireDate	Address	City	Region	PostalCode	Country	HomePhone	Extension	Photo
1	Davolio	Nancy	Sales Representative	Ms.	1948-12-08 00:00:00.000	1992-05-01 00:00:00.000	507 - 20th Ave. E. Apt. 2A	Seattle	WA	98122	USA	(206) 555-8657	5467	0x151C2F0002000000D000E001400210
2	Fuller	Andrew	Vice President, Sales	Dr.	1952-02-19 00:00:00.000	1992-08-14 00:00:00.000	908 W. Capital Way	Tacoma	WA	98401	USA	(206) 555-9482	3457	0x151C2F0002000000D000E001400210
3	Leverling	Janet	Sales Representative	Mrs.	1963-08-30 00:00:00.000	1992-04-01 00:00:00.000	722 Moss Bay Blvd.	Kirkland	WA	98033	USA	(206) 555-3412	3355	0x151C2F0002000000D000E001400210
4	Peacock	Margaret	Sales Representative	Mrs.	1937-09-12 00:00:00.000	1993-05-03 00:00:00.000	4110 Old Redmond Rd.	Redmond	WA	98052	USA	(206) 555-8122	5176	0x151C2F0002000000D000E001400210
5	Buchanan	Steven	Sales Manager	Mr.	1955-03-04 00:00:00.000	1993-10-17 00:00:00.000	14 Garrett Hill	London	NULL	SW1 8JR	UK	(71) 555-4848	3453	0x151C2F0002000000D000E001400210
6	Suyama	Michael	Sales Representative	Mr.	1963-07-02 00:00:00.000	1993-10-17 00:00:00.000	Coventry House, Miner Rd.	London	NULL	EC2 7JR	UK	(71) 555-7773	428	0x151C2F0002000000D000E001400210
7	King	Robert	Sales Representative	Mr.	1960-05-29 00:00:00.000	1994-01-02 00:00:00.000	Edgarham Hollow, Winchester Way	London	NULL	RG1 9SP	UK	(71) 555-5598	465	0x151C2F0002000000D000E001400210
8	Callahan	Laura	Inside Sales Coordinator	Ms.	1958-01-09 00:00:00.000	1994-03-05 00:00:00.000	4726 - 11th Ave. N.E.	Seattle	WA	98105	USA	(206) 555-1189	2344	0x151C2F0002000000D000E001400210
9	Dodsworth	Anne	Sales Representative	Ms.	1966-01-27 00:00:00.000	1994-11-15 00:00:00.000	7 Houndstooth Rd.	London	NULL	WG2 7LT	UK	(71) 555-4444	452	0x151C2F0002000000D000E001400210

**SELECT \* FROM Employees WHERE Title LIKE 'SALES%';**

Dit toont alle medewerkers waarvan de functie met het woord SALES begint.

EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	BirthDate	HireDate	Address	City	Region	PostalCode	Country	HomePhone	Extension	Photo
1	Davolio	Nancy	Sales Representative	Ms.	1948-12-08 00:00:00.000	1992-05-01 00:00:00.000	507 - 20th Ave. E. Apt. 2A	Seattle	WA	98122	USA	(206) 555-8657	5467	0x151C2F0002000000D000E001400210P
2	Leverling	Janet	Sales Representative	Mrs.	1963-08-30 00:00:00.000	1992-04-01 00:00:00.000	722 Moss Bay Blvd.	Kirkland	WA	98033	USA	(206) 555-3412	3355	0x151C2F0002000000D000E001400210P
3	Peacock	Margaret	Sales Representative	Mrs.	1937-09-19 00:00:00.000	1993-05-03 00:00:00.000	4110 Old Redmond Rd.	Redmond	WA	98052	USA	(206) 555-8122	5176	0x151C2F0002000000D000E001400210P
4	Buchanan	Steven	Sales Manager	Mr.	1955-03-04 00:00:00.000	1993-10-17 00:00:00.000	14 Garrett Hill	London	NULL	SW1 8JR	UK	(71) 555-4848	3453	0x151C2F0002000000D000E001400210P
5	Suyama	Michael	Sales Representative	Mr.	1963-07-02 00:00:00.000	1993-10-17 00:00:00.000	Coventry House, Miner Rd.	London	NULL	EC2 7JR	UK	(71) 555-7773	428	0x151C2F0002000000D000E001400210P
6	King	Robert	Sales Representative	Mr.	1958-05-29 00:00:00.000	1994-01-02 00:00:00.000	Edgarham Hollow, Winchester Way	London	NULL	RG1 9SP	UK	(71) 555-5598	465	0x151C2F0002000000D000E001400210P
7	Callahan	Laura	Inside Sales Coordinator	Ms.	1958-01-09 00:00:00.000	1994-03-05 00:00:00.000	4726 - 11th Ave. N.E.	Seattle	WA	98105	USA	(206) 555-1189	2344	0x151C2F0002000000D000E001400210P
8	Dodsworth	Anne	Sales Representative	Ms.	1966-01-27 00:00:00.000	1994-11-15 00:00:00.000	7 Houndstooth Rd.	London	NULL	WG2 7LT	UK	(71) 555-4444	452	0x151C2F0002000000D000E001400210P

**SELECT \* FROM Employees WHERE Title LIKE '%SALES';**

Dit toont alle medewerkers waarvan de functie met het woord SALES eindigt.

EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	BirthDate	HireDate	Address	City	Region	PostalCode	Country	HomePhone	Extension	Photo
1	Fuller	Andrew	Vice President, Sales	Dr.	1952-02-19 00:00:00.000	1992-08-14 00:00:00.000	908 W. Capital Way	Tacoma	WA	98401	USA	(206) 555-9482	3457	0x151C2F0002000000D000E001400210FFFFFF428746

Het is ook mogelijk om voor de string een keuze optie te maken door de opties tussen haken [] te plaatsen, bijvoorbeeld als de string moet beginnen met een letter V of met een letter I:

**SELECT \* FROM Employees WHERE Title LIKE ('[V,I]%'');**

Dit toont alle medewerkers waarvan de Title begint met een letter V of een letter I.

EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	BirthDate	HireDate	Address	City	Region	PostalCode
1	Fuller	Andrew	Vice President, Sales	Dr.	1952-02-19 00:00:00.000	1992-08-14 00:00:00.000	908 W. Capital Way	Tacoma	WA	98401
2	Callahan	Laura	Inside Sales Coordinator	Ms.	1958-01-09 00:00:00.000	1994-03-05 00:00:00.000	4726 - 11th Ave. N.E.	Seattle	WA	98105

## 8.10 ABS (NEW)

Met ABS kan de absolute waarde (positieve waarde) van een getal in een kolom worden berekend:

```
SELECT ProductID, UnitsInStock - ReorderLevel
FROM Products;
SELECT ProductID, ABS(UnitsInStock - ReorderLevel)
FROM Products;
```

Screenshot of SQL Server Management Studio showing the execution of the second query from the previous code block. The result set shows three rows of data. The third row, where ProductID is 3, has the value -12 in the 'UnitsInStock - ReorderLevel' column. A red box highlights this value, and a red arrow points from it to the corresponding value of 12 in the result set below, which is the output of the ABS function applied to the same row.

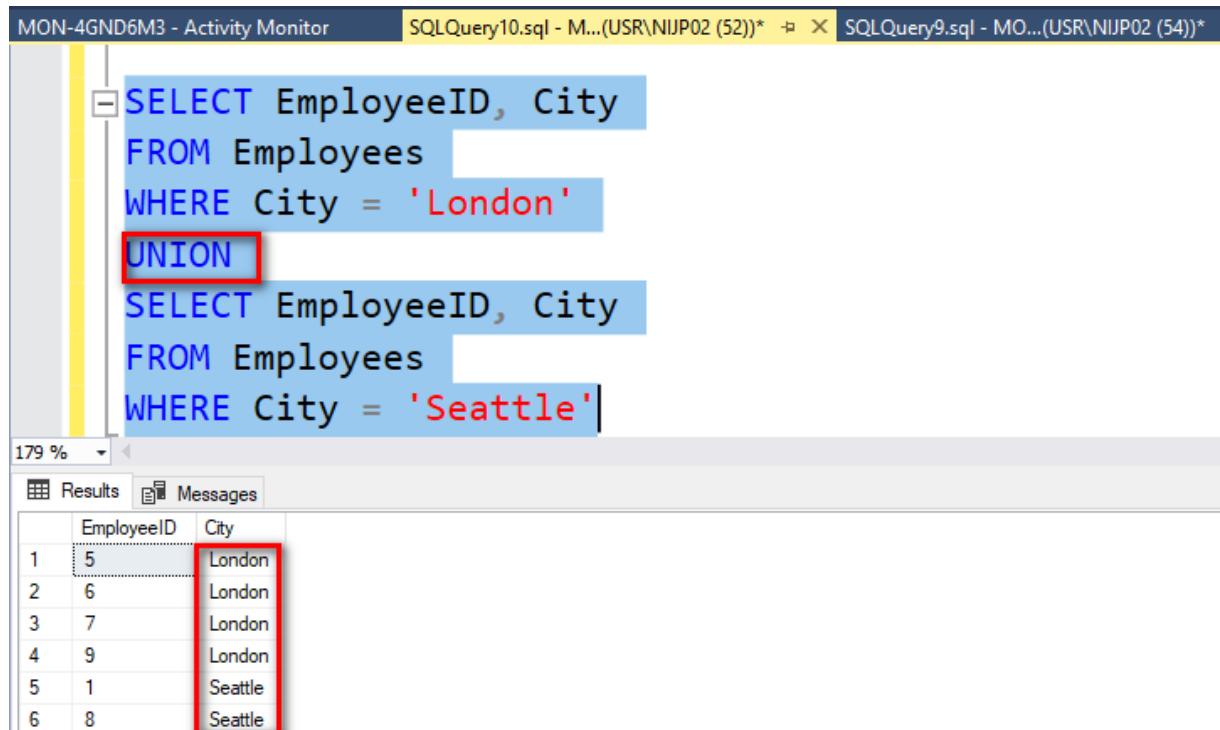
ProductID	(No column name)
1	29
2	-8
3	-12

ProductID	(No column name)
1	29
2	8
3	12

## 8.11 UNION (NEW)

Met UNION kunnen we de resultaten van twee tabelexpressies achter elkaar plakken. Het eindresultaat bevat elke rij die in het resultaat van één van de twee tabelexpressies of in beide voorkomt:

```
SELECT EmployeeID, City
FROM Employees
WHERE City = 'London'
UNION
SELECT EmployeeID, City
FROM Employees
WHERE City = 'Seattle'
```



The screenshot shows the SQL Server Management Studio interface. In the top tab bar, there are three tabs: 'MON-4GND6M3 - Activity Monitor', 'SQLQuery10.sql - M...(USR\NIJP02 (52))\*', and 'SQLQuery9.sql - MO...(USR\NIJP02 (54))\*'. The main window displays a T-SQL script with a red box highlighting the 'UNION' keyword. Below the script, the 'Results' tab is selected, showing a table with two columns: 'EmployeeID' and 'City'. The data in the table is as follows:

	EmployeeID	City
1	5	London
2	6	London
3	7	London
4	9	London
5	1	Seattle
6	8	Seattle

Voorwaarden voor UNION zijn dat de tabelexpressies vergelijkbare datatypen hebben en dat een ORDER BY pas na de laatste expressie op het gehele resultaat mag worden uitgevoerd. Door de UNION wordt automatisch de dubbele rijen uit het eindresultaat verwijderd, dus een DISTINCT is overbodig (maar wel toegestaan).

## 8.12 INTERSECT (NEW)

Met INTERSECT kunnen we de resultaten van twee tabelexpressies combineren, waarbij het eindresultaat alleen de rijen bevat die in de resultaten van beide tabelexpressies voorkomen.

```
SELECT EmployeeID, City, BirthDate
FROM Employees
WHERE City = 'London'
INTERSECT
SELECT EmployeeID, City, BirthDate
FROM Employees
WHERE BirthDate = '1955-03-04 00:00:00.000';
```

MON-4GND6M3 - Activity Monitor    SQLQuery10.sql - M...(USR\NIJP02 (52))\*    SQLQuery9.sql - MO...(USR)

```

SELECT EmployeeID, City, BirthDate
FROM Employees
WHERE City = 'London'
INTERSECT
SELECT EmployeeID, City, BirthDate
FROM Employees
WHERE BirthDate = '1955-03-04 00:00:00.000';

```

179 %

	EmployeeID	City	BirthDate
1	5	London	1955-03-04 00:00:00.000

Dubbele rijen worden net als bij de UNION automatisch uit het resultaat verwijderd. Voor de INTERSECT gelden dezelfde voorwaarden als bij de UNION.

### 8.13 EXCEPT (NEW)

Met EXCEPT kunnen we twee tabelexpressies combineren, waarbij het eindresultaat allen de rijen bevat die wel in het resultaat van de eerste tabel expressie voorkomen maar niet in het resultaat van de tweede tabel expressie voorkomen.

MON-4GND6M3 - Activity Monitor    SQLQuery10.sql - M...(USR\NIJP02 (52))\*    SQLQuery9.sql - MO...(USR)

```

SELECT EmployeeID, City, BirthDate
FROM Employees
WHERE City = 'London'
EXCEPT
SELECT EmployeeID, City, BirthDate
FROM Employees
WHERE BirthDate = '1955-03-04 00:00:00.000'
ORDER BY EmployeeID;

```

179 %

	EmployeeID	City	BirthDate
1	6	London	1963-07-02 00:00:00.000
2	7	London	1960-05-29 00:00:00.000
3	9	London	1966-01-27 00:00:00.000

Dubbele rijen worden net zoals bij de UNION en de INTERSECT automatisch verwijderd. Voor de EXCEPT geldt dezelfde voorwaarden als bij de UNION en de INTERSECT.

## 8.14 UNION ALL, INTERSECT ALL en EXCEPT ALL (NEW)

Door ALL toe te voegen achter de UNION, INTERSECT of EXCEPT kunnen we het verwijderen van dubbele rijen onderdrukken. De dubbele rijen worden door de ALL dan niet verwijderd.

## 8.15 NULL waarden bij UNION, INTERSECT en EXCEPT (NEW)

Wanneer er NULL waarden in beide tabellexpressies zitten dan beschouwt SQL de NULL waarden als gelijk en komt er slechts één NULL waarde in het eindresultaat.

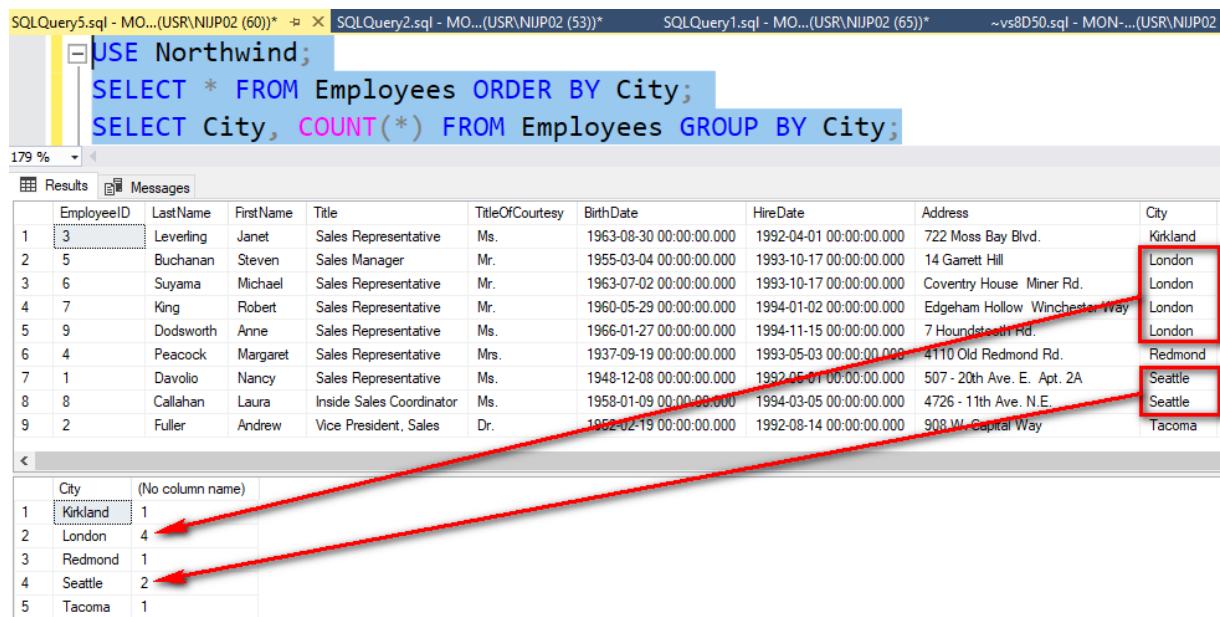
## 8.16 Meer dan 2 UNIONS, INTERSECTS en EXCEPTS (NEW)

Wanneer er meerdere UNION, INTERSECT en/of EXCEPTs worden gebruikt dan is de werkwijze van boven naar beneden, dus eerst de combinatie van de eerste en de tweede tabellexpressie en daarna het resultaat daarvan gecombineerd met de volgende tabellexpressie.

## 8.17 GROUP BY (NEW)

Met GROUP BY kunnen we rijen groeperen op basis van onderlinge overeenkomsten. We kunnen bijvoorbeeld de medewerkers groeperen op basis van woonplaats en dan per woonplaats opvragen hoeveel medewerkers er in die woonplaats wonen. De GROUP BY produceert dan per groep één rij met de gesommeerde gegevens. Voor elke gegroepeerde rij wordt nu een COUNT(\*) berekend.

```
USE Northwind;
SELECT * FROM Employees ORDER BY City;
SELECT City, COUNT(*) FROM Employees GROUP BY City;
```



	EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	BirthDate	HireDate	Address	City
1	3	Leverling	Janet	Sales Representative	Ms.	1963-08-30 00:00:00.000	1992-04-01 00:00:00.000	722 Moss Bay Blvd.	Kirkland
2	5	Buchanan	Steven	Sales Manager	Mr.	1955-03-04 00:00:00.000	1993-10-17 00:00:00.000	14 Garrett Hill	London
3	6	Suyama	Michael	Sales Representative	Mr.	1963-07-02 00:00:00.000	1993-10-17 00:00:00.000	Coventry House, Miner Rd.	London
4	7	King	Robert	Sales Representative	Mr.	1960-05-29 00:00:00.000	1994-01-02 00:00:00.000	Edgeham Hollow, Winchester Way	London
5	9	Dodsworth	Anne	Sales Representative	Ms.	1966-01-27 00:00:00.000	1994-11-15 00:00:00.000	7 Houndsworth Rd.	London
6	4	Peacock	Margaret	Sales Representative	Mrs.	1937-09-19 00:00:00.000	1993-05-03 00:00:00.000	4110 Old Redmond Rd.	Redmond
7	1	Davolio	Nancy	Sales Representative	Ms.	1948-12-08 00:00:00.000	1992-05-01 00:00:00.000	507 - 20th Ave. E. Apt. 2A	Seattle
8	8	Callahan	Laura	Inside Sales Coordinator	Ms.	1958-01-09 00:00:00.000	1994-03-05 00:00:00.000	4726 - 11th Ave. N.E.	Seattle
9	2	Fuller	Andrew	Vice President, Sales	Dr.	1952-02-19 00:00:00.000	1992-08-14 00:00:00.000	908 W. Capital Way	Tacoma

	City	(No column name)
1	Kirkland	1
2	London	4
3	Redmond	1
4	Seattle	2
5	Tacoma	1

We hebben nu gegroepeerd op kolom City, maar we kunnen ook groeperen op twee of meerdere kolommen. Zo kunnen we bijvoorbeeld groeperen op functie en woonplaats.

```
SELECT * FROM Employees ORDER BY Title, City;
SELECT Title, City, COUNT(*) FROM Employees GROUP BY Title, City;
```

SQLQuery5.sql - MO... (USR\NIJP02 (60)) \* SQLQuery2.sql - MO... (USR\NIJP02 (53)) \* SQLQuery1.sql - MO... (USR\NIJP02 (65)) \* ~vs8D50.sql - MON... (USR\NIJP02)

```

SELECT * FROM Employees ORDER BY Title, City;

SELECT Title, City, COUNT(*) FROM Employees GROUP BY Title, City;

```

179 %

	EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	BirthDate	HireDate	Address	City
1	8	Callahan	Laura	Inside Sales Coordinator	Ms.	1958-01-09 00:00:00.000	1994-03-05 00:00:00.000	4726 - 11th Ave. N.E.	Seattle
2	5	Buchanan	Steven	Sales Manager	Mr.	1955-03-04 00:00:00.000	1993-10-17 00:00:00.000	14 Garrett Hill	London
3	3	Leverling	Janet	Sales Representative	Ms.	1963-08-30 00:00:00.000	1992-04-01 00:00:00.000	722 Moss Bay Blvd.	Kirkland
4	6	Suyama	Michael	Sales Representative	Mr.	1963-07-02 00:00:00.000	1993-10-17 00:00:00.000	Coventry House Miner Rd.	London
5	7	King	Robert	Sales Representative	Mr.	1960-05-29 00:00:00.000	1994-01-02 00:00:00.000	Edgeham Hollow Winchester Way	London
6	9	Dodsworth	Anne	Sales Representative	Ms.	1966-01-27 00:00:00.000	1994-11-15 00:00:00.000	7 Houndstooth Rd.	London
7	4	Peacock	Margaret	Sales Representative	Mrs.	1937-09-19 00:00:00.000	1993-05-03 00:00:00.000	4110 Old Redmond Rd.	Redmond
8	1	Davolio	Nancy	Sales Representative	Ms.	1948-12-08 00:00:00.000	1992-05-01 00:00:00.000	507 - 20th Ave. E. Apt. 2A	Seattle
9	2	Fuller	Andrew	Vice President, Sales	Dr.	1952-02-19 00:00:00.000	1992-08-14 00:00:00.000	908 W. Capital Way	Tacoma

	Title	City	(No column name)
1	Sales Representative	Kirkland	1
2	Sales Manager	London	1
3	Sales Representative	London	3
4	Sales Representative	Redmond	1
5	Inside Sales Coordinator	Seattle	1
6	Sales Representative	Seattle	1
7	Vice President, Sales	Tacoma	1

De volgorde van de kolommen in de GROUP BY heeft geen effect op het eindresultaat van een instructie. Groeperen op de kolommen Title, City heeft hetzelfde resultaat als groeperen op City, Title.

Behalve op kolommen kunnen we ook groeperen op **Expressie** (SQL-instructie). Zo kunnen we groeperen op het jaar dat medewerkers in dienst zijn gekomen. Met de expressie YEAR(HireDate) halen we het jaartal uit de kolom HireDate. Rijen waarvoor de waarde van de expressie YEAR(HireDate) gelijk is, vormen hierna een groep.

```

SELECT * FROM Employees ORDER BY HireDate;
SELECT YEAR(HireDate) AS Jaartal, COUNT(*) FROM Employees
GROUP BY YEAR(HireDate);

```

SQLQuery5.sql - MO... (USR\NIJP02 (60)) \* SQLQuery2.sql - MO... (USR\NIJP02 (53)) \* SQLQuery1.sql - MO... (USR\NIJP02 (65)) \* ~vs8D50.sql - MON... (USR\NIJP02)

```

SELECT * FROM Employees ORDER BY HireDate;
SELECT YEAR(HireDate) AS Jaartal, COUNT(*) FROM Employees
GROUP BY YEAR(HireDate);

```

179 %

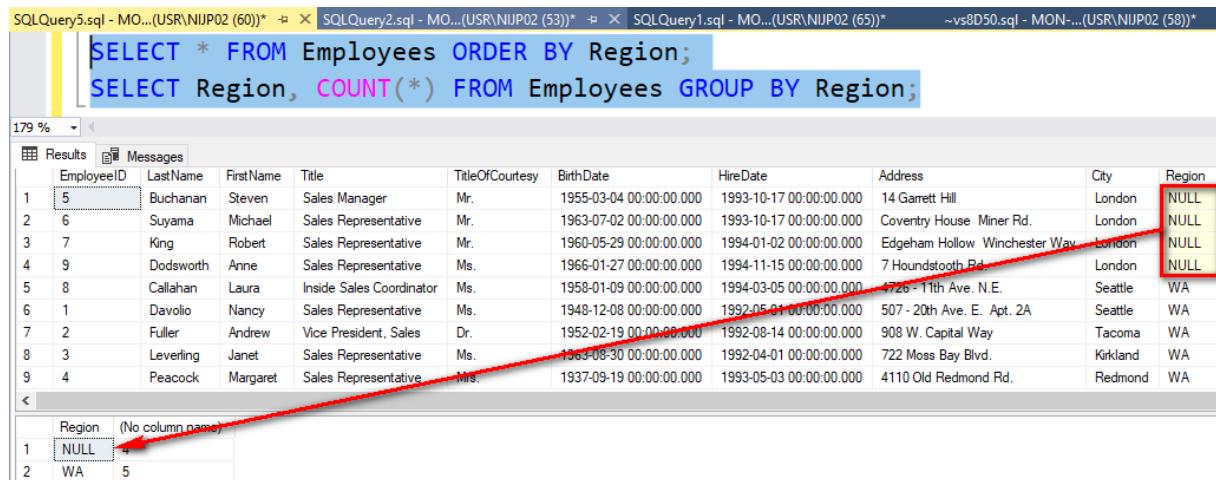
	EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	BirthDate	HireDate	Address	City
1	3	Leverling	Janet	Sales Representative	Ms.	1963-08-30 00:00:00.000	1992-04-01 00:00:00.000	722 Moss Bay Blvd.	Kirkland
2	1	Davolio	Nancy	Sales Representative	Ms.	1948-12-08 00:00:00.000	1992-05-01 00:00:00.000	507 - 20th Ave. E. Apt. 2A	Seattle
3	2	Fuller	Andrew	Vice President, Sales	Dr.	1952-02-19 00:00:00.000	1992-08-14 00:00:00.000	908 W. Capital Way	Tacoma
4	4	Peacock	Margaret	Sales Representative	Mrs.	1937-09-19 00:00:00.000	1993-05-03 00:00:00.000	4110 Old Redmond Rd.	Redmond
5	5	Buchanan	Steven	Sales Manager	Mr.	1955-03-04 00:00:00.000	1993-10-17 00:00:00.000	14 Garrett Hill	London
6	6	Suyama	Michael	Sales Representative	Mr.	1963-07-02 00:00:00.000	1993-10-17 00:00:00.000	Coventry House Miner Rd.	London
7	7	King	Robert	Sales Representative	Mr.	1960-05-29 00:00:00.000	1994-01-02 00:00:00.000	Edgeham Hollow Winchester Way	London
8	8	Callahan	Laura	Inside Sales Coordinator	Ms.	1958-01-09 00:00:00.000	1994-03-05 00:00:00.000	4726 - 11th Ave. N.E.	Seattle
9	9	Dodsworth	Anne	Sales Representative	Ms.	1966-01-27 00:00:00.000	1994-11-15 00:00:00.000	7 Houndstooth Rd.	London

	Jaartal	(No column name)
1	1992	3
2	1993	3
3	1994	3

De expressies waarop gegroepeerd wordt mag complex zijn en mag ook uit subquery's bestaan.

Wanneer we groeperen op een kolom waarin NULL-waarden voorkomen, vormen alle NULL-waarden één groep.



```

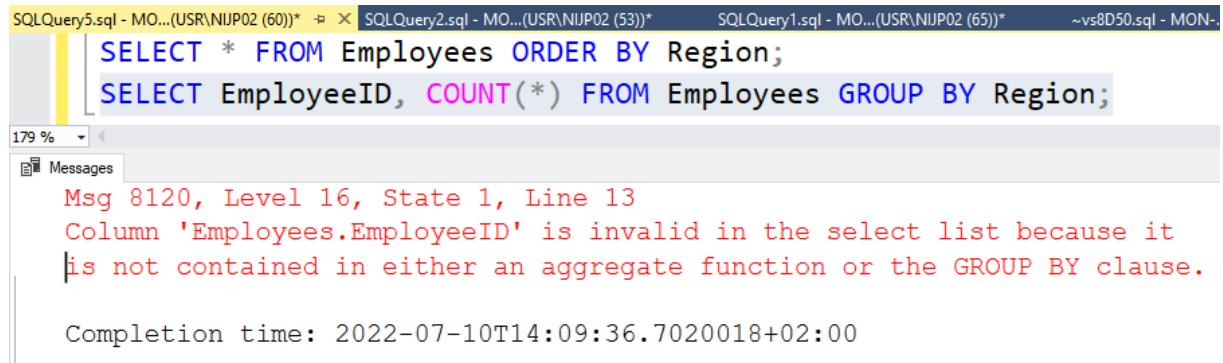
SQLQuery5.sql - MO...((USR\NIJP02 (60))*
SELECT * FROM Employees ORDER BY Region;
SELECT Region, COUNT(*) FROM Employees GROUP BY Region;
    
```

	EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	BirthDate	HireDate	Address	City	Region
1	5	Buchanan	Steven	Sales Manager	Mr.	1955-03-04 00:00:00.000	1993-10-17 00:00:00.000	14 Garrett Hill	London	NULL
2	6	Suyama	Michael	Sales Representative	Mr.	1963-07-02 00:00:00.000	1993-10-17 00:00:00.000	Coventry House Miner Rd.	London	NULL
3	7	King	Robert	Sales Representative	Mr.	1960-05-29 00:00:00.000	1994-01-02 00:00:00.000	Edgeham Hollow Winchester Way	London	NULL
4	9	Dodsworth	Anne	Sales Representative	Ms.	1966-01-27 00:00:00.000	1994-11-15 00:00:00.000	7 Houndstooth Rd.	London	NULL
5	8	Callahan	Laura	Inside Sales Coordinator	Ms.	1958-01-09 00:00:00.000	1994-03-05 00:00:00.000	4726 - 11th Ave. N.E.	Seattle	WA
6	1	Davolio	Nancy	Sales Representative	Ms.	1948-12-08 00:00:00.000	1992-05-01 00:00:00.000	507 - 20th Ave. E. Apt. 2A	Seattle	WA
7	2	Fuller	Andrew	Vice President, Sales	Dr.	1952-02-19 00:00:00.000	1992-08-14 00:00:00.000	908 W. Capital Way	Tacoma	WA
8	3	Leverling	Janet	Sales Representative	Ms.	1963-08-30 00:00:00.000	1992-04-01 00:00:00.000	722 Moss Bay Blvd.	Kirkland	WA
9	4	Peacock	Margaret	Sales Representative	Mrs.	1937-09-19 00:00:00.000	1993-05-03 00:00:00.000	4110 Old Redmond Rd.	Redmond	WA

	Region	(No column name)
1	NULL	4
2	WA	5

Een kolom waarop niet gegroepeerd is kan niet getoond worden, anders ontstaat er een foutmelding.



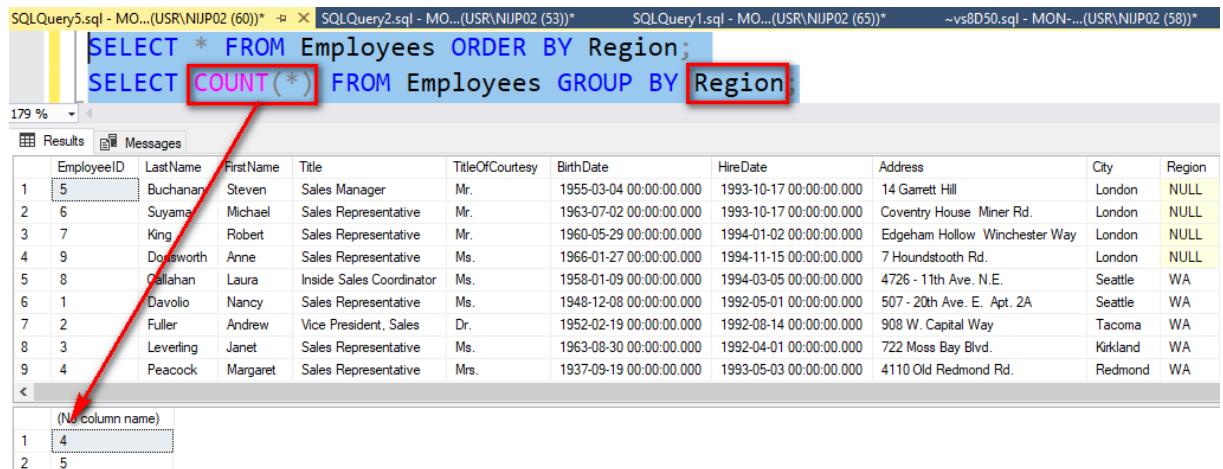
```

SQLQuery5.sql - MO...((USR\NIJP02 (60))*
SELECT * FROM Employees ORDER BY Region;
SELECT EmployeeID, COUNT(*) FROM Employees GROUP BY Region;
    
```

Msg 8120, Level 16, State 1, Line 13  
Column 'Employees.EmployeeID' is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause.

Completion time: 2022-07-10T14:09:36.7020018+02:00

Meestal staat de expressie waarop gegroepeerd wordt ook vermeld in de SELECT, maar dat is niet noodzakelijk.



```

SQLQuery5.sql - MO...((USR\NIJP02 (60))*
SELECT * FROM Employees ORDER BY Region;
SELECT COUNT(*) FROM Employees GROUP BY Region;
    
```

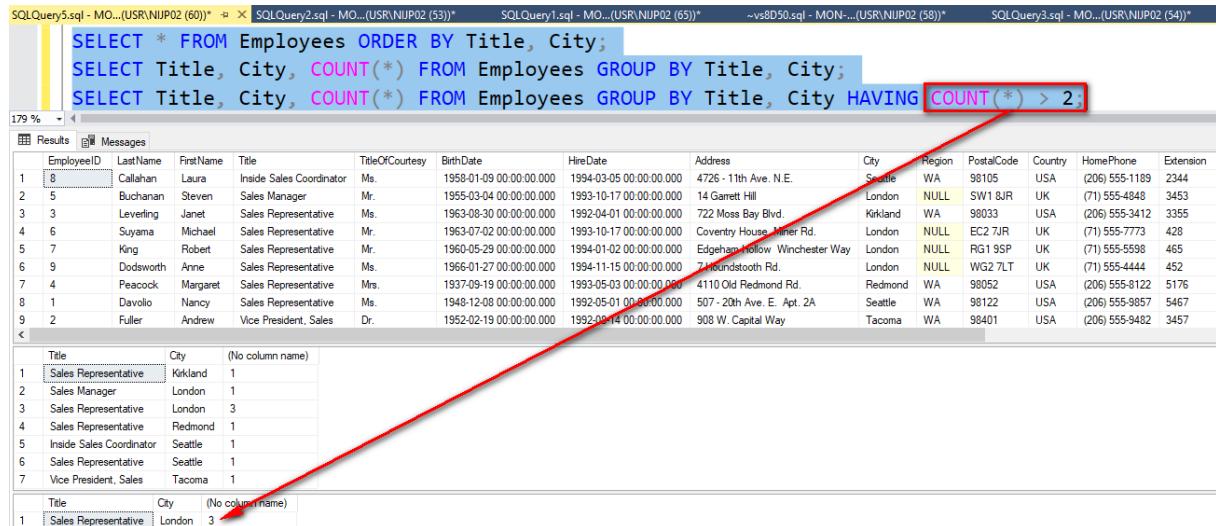
	EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	BirthDate	HireDate	Address	City	Region
1	5	Buchanan	Steven	Sales Manager	Mr.	1955-03-04 00:00:00.000	1993-10-17 00:00:00.000	14 Garrett Hill	London	NULL
2	6	Suyama	Michael	Sales Representative	Mr.	1963-07-02 00:00:00.000	1993-10-17 00:00:00.000	Coventry House Miner Rd.	London	NULL
3	7	King	Robert	Sales Representative	Mr.	1960-05-29 00:00:00.000	1994-01-02 00:00:00.000	Edgeham Hollow Winchester Way	London	NULL
4	9	Dodsworth	Anne	Sales Representative	Ms.	1966-01-27 00:00:00.000	1994-11-15 00:00:00.000	7 Houndstooth Rd.	London	NULL
5	8	Callahan	Laura	Inside Sales Coordinator	Ms.	1958-01-09 00:00:00.000	1994-03-05 00:00:00.000	4726 - 11th Ave. N.E.	Seattle	WA
6	1	Davolio	Nancy	Sales Representative	Ms.	1948-12-08 00:00:00.000	1992-05-01 00:00:00.000	507 - 20th Ave. E. Apt. 2A	Seattle	WA
7	2	Fuller	Andrew	Vice President, Sales	Dr.	1952-02-19 00:00:00.000	1992-08-14 00:00:00.000	908 W. Capital Way	Tacoma	WA
8	3	Leverling	Janet	Sales Representative	Ms.	1963-08-30 00:00:00.000	1992-04-01 00:00:00.000	722 Moss Bay Blvd.	Kirkland	WA
9	4	Peacock	Margaret	Sales Representative	Mrs.	1937-09-19 00:00:00.000	1993-05-03 00:00:00.000	4110 Old Redmond Rd.	Redmond	WA

	(No column name)
1	4
2	5

## 8.18 HAVING (NEW)

De HAVING heeft een functie die vergelijkbaar is met die van de WHERE. Het verschil is dat met WHERE rijen geselecteerd worden nadat de FROM verwerkt is, terwijl met de HAVING rijen geselecteerd worden nadat een GROUP BY uitgevoerd is. In tegenstelling tot de WHERE mag in de HAVING wél aggregatiefuncties worden gebruikt, zoals bijvoorbeeld COUNT(\*).



```

SQLQuery5.sql - MO...\\USR\\NJP02 (60)*  SQLQuery2.sql - MO...\\USR\\NJP02 (53)*  SQLQuery1.sql - MO...\\USR\\NJP02 (65)*  ~vs8D50.sql - MON...\\USR\\NJP02 (58)*  SQLQuery3.sql - MO...\\USR\\NJP02 (54)*

SELECT * FROM Employees ORDER BY Title, City;
SELECT Title, City, COUNT(*) FROM Employees GROUP BY Title, City;
SELECT Title, City, COUNT(*) FROM Employees GROUP BY Title, City HAVING COUNT(*) > 2;
    
```

EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	BirthDate	HireDate	Address	City	Region	PostalCode	Country	HomePhone	Extension
1	Callahan	Laura	Inside Sales Coordinator	Ms.	1959-01-09 00:00:00.000	1994-03-05 00:00:00.000	4726 - 11th Ave. N.E.	Seattle	WA	98105	USA	(206) 555-1189	2344
2	Buchanan	Steven	Sales Manager	Mr.	1955-03-04 00:00:00.000	1993-10-17 00:00:00.000	14 Garrett Hill	London	NULL	SW1 8JR	UK	(71) 555-4848	3453
3	Levin	Janet	Sales Representative	Ms.	1963-08-30 00:00:00.000	1992-04-01 00:00:00.000	722 Moss Bay Blvd.	Kirkland	WA	98033	USA	(206) 555-3412	3355
4	Suyama	Michael	Sales Representative	Mr.	1963-07-02 00:00:00.000	1993-10-17 00:00:00.000	Coventry House, Inner Rd.	London	NULL	EC2 7JR	UK	(71) 555-7773	428
5	King	Robert	Sales Representative	Mr.	1960-05-29 00:00:00.000	1994-01-02 00:00:00.000	Edgeham Hollow, Winchester Way	London	NULL	RG1 9SP	UK	(71) 555-5598	465
6	Dodsworth	Anne	Sales Representative	Ms.	1966-01-27 00:00:00.000	1994-11-15 00:00:00.000	71 London St.	London	NULL	WG2 7LT	UK	(71) 555-4444	452
7	Peacock	Margaret	Sales Representative	Mrs.	1937-09-19 00:00:00.000	1993-05-03 00:00:00.000	4110 Old Redmond Rd.	Redmond	WA	98052	USA	(206) 555-8122	5176
8	Davolio	Nancy	Sales Representative	Ms.	1948-12-08 00:00:00.000	1992-05-01 00:00:00.000	507 - 20th Ave. E. Apt. 2A	Seattle	WA	98122	USA	(206) 555-9857	5467
9	Fuller	Andrew	Vice President, Sales	Dr.	1952-02-19 00:00:00.000	1992-09-14 00:00:00.000	908 W. Capital Way	Tacoma	WA	98401	USA	(206) 555-9482	3457

Title	City	(No column name)
Sales Representative	Kirkland	1
Sales Manager	London	1
Sales Representative	London	3
Sales Representative	Redmond	1
Inside Sales Coordinator	Seattle	1
Sales Representative	Seattle	1
Vice President, Sales	Tacoma	1

Title	City	(No column name)
Sales Representative	London	3

## 8.19 COMMON TABLE EXPRESSION CTE (NEW)

Common Table Expressions CTE zijn een standaardvorm van tabeluitdrukkingen die vergelijkbaar is met afgeleide tabellen met een aantal belangrijke voordelen. Common Table Expressions worden gedefinieerd met de syntax:

**WITH <CTE naam> AS (<CTE innerquery>) <CTE outerquery>;**

Een bestelling (Order) bij Northwind Traders bestaat uit een aantal producten (Order Detail), waarvan er per product meerdere exemplaren zijn besteld. De klant kan bij Northwind een klantenkorting krijgen op verschillende producten. Om nu de totale kosten van de bestelling te berekenen moeten we eerst de prijs per productregel berekenen (horizontaal) en daarna alle totalen van de producten bij elkaar optellen (verticaal).

Met GROUP BY kunnen we, zoals beschreven in de vorige paragraaf, aggregatiefuncties gebruiken in het SELECT statement. We kunnen in het resultaat een kolom Total toevoegen waarin de prijs per unit vermenigvuldigt is met het bestelde aantal en verminderd met de klantkorting (Discount).

```

SELECT * FROM [dbo].[Order Details] WHERE OrderID = 11071;
SELECT [OrderID], [ProductID], [UnitPrice], [Quantity], [Discount],
SUM([UnitPrice] * [Quantity]) * (1 - Discount) AS Total
FROM [dbo].[Order Details] WHERE OrderID = 11071
GROUP BY [OrderID], [ProductID], [UnitPrice], [Quantity], [Discount];
    
```

SQLQuery5.sql - MO... (USR\NIJP02 (60)) \* SQLQuery2.sql - MO... (USR\NIJP02 (53)) \* SQLQuery1.sql - MO... (USR\NIJP02 (65)) \* ~vs8D50.sql - MON...

```

SELECT * FROM [dbo].[Order Details] WHERE OrderID = 11071;
SELECT [OrderID], [ProductID], [UnitPrice], [Quantity], [Discount],
SUM([UnitPrice] * [Quantity]) * (1 - Discount) AS Total
FROM [dbo].[Order Details] WHERE OrderID = 11071
GROUP BY [OrderID], [ProductID], [UnitPrice], [Quantity], [Discount];

```

Results

OrderID	ProductID	UnitPrice	Quantity	Discount	Total
1	11071	7	30.00	15	0.05
2	11071	13	6.00	10	0.05

OrderID	ProductID	UnitPrice	Quantity	Discount	Total
1	11071	7	30.00	15	427.5
2	11071	13	6.00	10	57

Om nu de totale kosten van de bestelling met het OrderID 11071 te berekenen kunnen we echter geen (verticale) SUM om de reeds gebruikte (horizontale) SUM toepassen, maar kunnen we gebruik maken van een Common Table Expression.

```

WITH CTE_ORDERS AS
(
SELECT [OrderID], [ProductID], [UnitPrice], [Quantity], [Discount],
SUM([UnitPrice] * [Quantity]) * (1 - Discount) AS Total
FROM [dbo].[Order Details] WHERE OrderID = 11071
GROUP BY [OrderID], [ProductID], [UnitPrice], [Quantity], [Discount]
)
SELECT SUM(Total) FROM CTE_ORDERS;

```

SQLQuery5.sql - MO... (USR\NIJP02 (60)) \* SQLQuery2.sql - MO... (USR\NIJP02 (53)) \* SQLQuery1.sql - MO... (USR\NIJP02 (65)) \* ~vs8D50.sql - MON...

```

SELECT [OrderID], [ProductID], [UnitPrice], [Quantity], [Discount],
SUM([UnitPrice] * [Quantity]) * (1 - Discount) AS Total
FROM [dbo].[Order Details] WHERE OrderID = 11071
GROUP BY [OrderID], [ProductID], [UnitPrice], [Quantity], [Discount];

WITH CTE_ORDERS AS
(
SELECT [OrderID], [ProductID], [UnitPrice], [Quantity], [Discount],
SUM([UnitPrice] * [Quantity]) * (1 - Discount) AS Total
FROM [dbo].[Order Details] WHERE OrderID = 11071
GROUP BY [OrderID], [ProductID], [UnitPrice], [Quantity], [Discount]
)
SELECT SUM(Total) AS GrandTotal FROM CTE_ORDERS;

```

Results

OrderID	ProductID	UnitPrice	Quantity	Discount	Total
1	11071	7	30.00	15	427.5
2	11071	13	6.00	10	57

GrandTotal
484.5

Met de GROUP BY hebben we dus horizontaal de kosten per product berekend en met de Common Table Expression hebben we verticaal de totale kosten van alle producten in de bestelling berekend.

## Hoofdstuk 9 CREATE,INSERT,UPDATE,ALTER,DELETE, DROP, Scripts

In de vorige hoofdstukken hebben we gekeken naar het selecteren van gegevens uit een tabel. In dit hoofdstuk gaan we kijken naar de SQL-statements om een database aan te maken, een tabel aan te maken, gegevens in te voeren, gegevens aan te passen en gegevens te verwijderen.

### 9.1 CREATE DATABASE en USE <databasenaam>

De SQL syntax om een database aan te maken is:

**CREATE DATABASE <databasenaam>;**

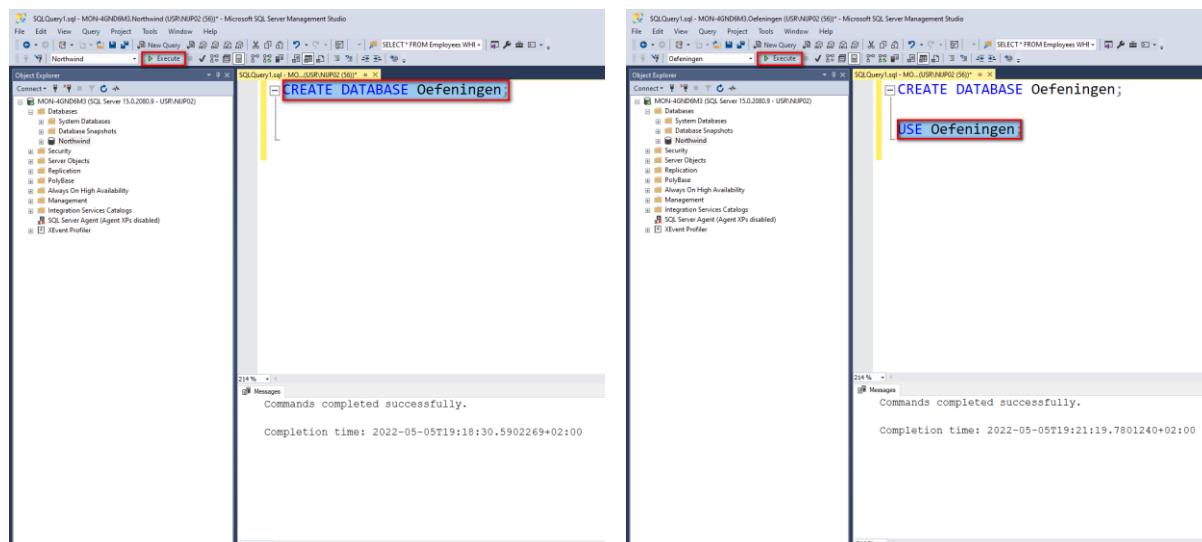
Nadat een database is aangemaakt kan de nieuwe database courant worden gemaakt met het commando:

**USE <databasenaam>;**

Voorbeeld:

**CREATE DATABASE Oefeningen;**

**USE Oefeningen;**



### 9.2 CREATE TABLE (NEW)

Nadat de nieuwe database courant is gemaakt kunnen we een tabel in de nieuwe database aanmaken.

De SQL syntax voor het aanmaken van een tabel is:

**CREATE TABLE <tabelnaam> (<kolomnaam1> <datatype>, <kolomnaam2> <datatype>, ...);**

Datatypen zijn het soort van gegevens dat in de kolom kan worden ingevoerd.

In de komende hoofdstukken zullen we ons slechts met een klein aantal datatypen bezig houden:

- INT = Integers (gehele getallen)
- REAL = Reële getallen (met decimale punt)
- VARCHAR(N) = string met een variabele lengte, met maximale lengte van N karakters

- DATE = datum YYYY-MM-DD
  - DATETIME = datum YYYY-MM-DD en tijd HH:MM:SS[.NNN]

Er bestaan vele andere datatypen. Voor meer informatie over de datatypen, zie:

<https://docs.microsoft.com/en-us/sql/t-sql/data-types/data-types-transact-sql?view=sql-server-ver15>

<https://www.sqlservertutorial.net/sql-server-basics/sql-server-data-types/>

### Voorbeeld:

```
CREATE TABLE Tabel01 (Naam VARCHAR(25), Leeftijd INT);
```

```
SELECT * FROM Tabel01;
```

The screenshot displays two instances of Microsoft SQL Server Management Studio (SSMS) running side-by-side. Both instances are connected to the same database, 'Oefeningen', located on the server 'M01-AS001\SQL Server 15.0.2086.3 (SP1 CU10) [USR-NPFR]'.  
  
Left Window (Screenshot 1): This window shows the Transact-SQL code being executed:  

```
CREATE DATABASE Oefeningen;
USE Oefeningen;
CREATE TABLE Tabel01 (Naam VARCHAR(25), Leeftijd INT);
```

  
The status bar at the bottom indicates:  

Commands completed successfully.  
Completion time: 2022-05-05T19:22:46.8066434+02:00

  
  
Right Window (Screenshot 2): This window shows the results of the query 'SELECT \* FROM Tabel01' being run:  

```
SELECT * FROM Tabel01;
```

  
The status bar at the bottom indicates:  

Query executed successfully.

### 9.3 DATE, DATETIME datatype en functie GETDATE (NEW)

De DATE en DATETIME datatypen kunnen we m.b.v. de functies YEAR, MONTH en DAY in het gewenste formaat opvragen, b.v.:

```
SELECT EmployeeID, FirstName, LastName, BirthDate FROM Employees;
SELECT EmployeeID, FirstName, LastName, YEAR(BirthDate) FROM Employees;
SELECT EmployeeID, FirstName, LastName, MONTH(BirthDate) FROM Employees;
SELECT EmployeeID, FirstName, LastName, DAY(BirthDate) FROM Employees;
```

MON-4GND6M3 - Activity Monitor SQLQuery10.sql - M...(\USR\NIJP02 (52))\* X SQLQuery9.sql - MO...(\USR\NIJP02 (54))\* SQLQuery8.sql - MO...(\USR\NIJP02 (55))

```
SELECT EmployeeID, FirstName, LastName, BirthDate FROM Employees;
SELECT EmployeeID, FirstName, LastName, YEAR(BirthDate) FROM Employees;
SELECT EmployeeID, FirstName, LastName, MONTH(BirthDate) FROM Employees;
SELECT EmployeeID, FirstName, LastName, DAY(BirthDate) FROM Employees;
```

179 %

Results Messages

	EmployeeID	FirstName	LastName	BirthDate
1	1	Nancy	Davolio	1948-12-08 00:00:00.000
2	2	Andrew	Fuller	1952-02-19 00:00:00.000
3	3	Janet	Leverling	1963-08-30 00:00:00.000

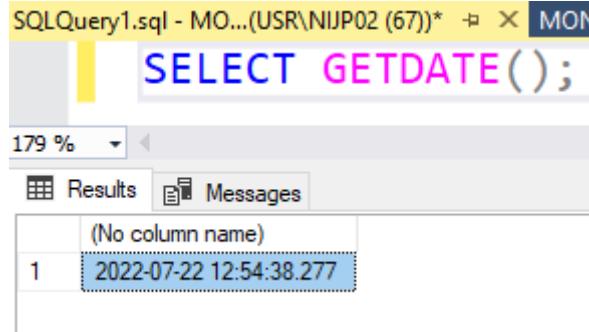
	EmployeeID	FirstName	LastName	(No column name)
1	1	Nancy	Davolio	1948
2	2	Andrew	Fuller	1952
3	3	Janet	Leverling	1963

	EmployeeID	FirstName	LastName	(No column name)
1	1	Nancy	Davolio	12
2	2	Andrew	Fuller	2
3	3	Janet	Leverling	8

	EmployeeID	FirstName	LastName	(No column name)
1	1	Nancy	Davolio	8
2	2	Andrew	Fuller	19
3	3	Janet	Leverling	30

Met de functie **GETDATE()** kunnen we de huidige datum en tijd opvragen, b.v.:

```
SELECT GETDATE();
```



The screenshot shows a SQL Server Management Studio window titled "SQLQuery1.sql - MON...". The query "SELECT GETDATE();" is run, and the result is displayed in the "Results" tab. The output is a single row with one column, showing the current date and time: 2022-07-22 12:54:38.277.

## 9.4 INSERT INTO

Nadat de tabel is aangemaakt kunnen we gegevens in de tabel invoeren. Het invoeren van gegevens in de tabel kan op twee manieren. De eerste manier is door de kolomnamen van de kolommen die je wilt invoeren expliciet te benoemen. De SQL syntax hiervoor is:

```
INSERT INTO <tabelnaam> (Kolomnaam1, Kolomnaam2,...) VALUES (<waarde1>,<waarde2>,...);
```

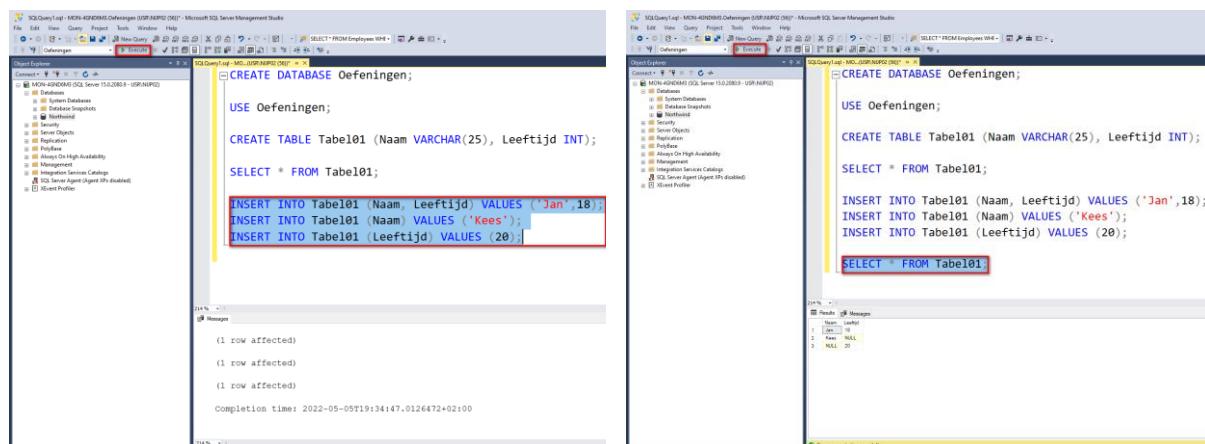
Hierbij is het mogelijk om slechts één of enkele kolommen die explicet zijn benoemd in te voeren en de overige kolommen leeg te laten.

De tweede manier is om de kolomnamen weg te laten, wat betekent dat je alle kolommen van de tabel wilt invoeren. De SQL syntax hiervoor is:

```
INSERT INTO <tabelnaam> VALUES (<waarde1>,<waarde2>,...);
```

Voorbeeld:

```
INSERT INTO Tabel01 (Naam, Leeftijd) VALUES ('Jan',18);
INSERT INTO Tabel01 (Naam) VALUES ('Kees');
INSERT INTO Tabel01 (Leeftijd) VALUES (20);
SELECT * FROM Tabel01;
```



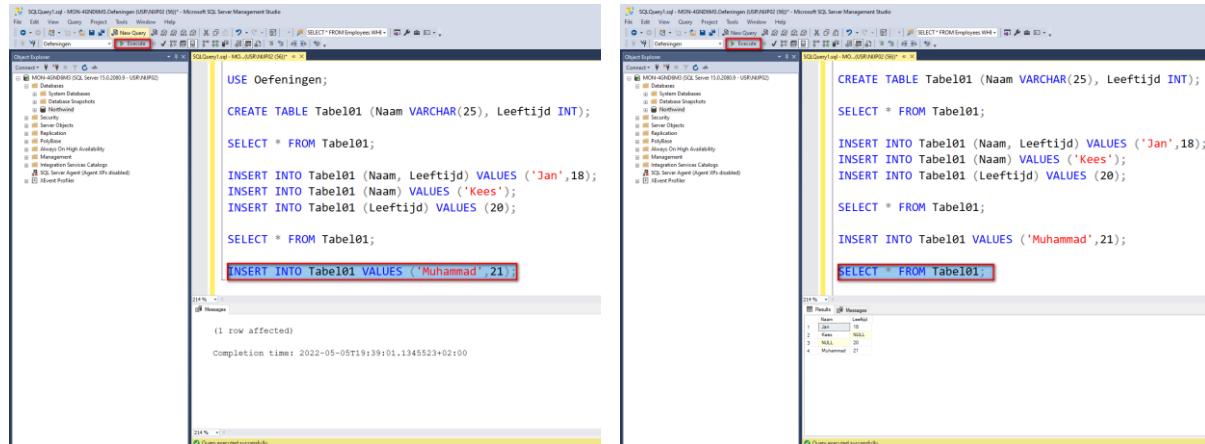
The screenshot shows two Microsoft SQL Server Management Studio windows. Both windows have the same script and execution results. The left window shows the explicit syntax: "INSERT INTO Tabel01 (Naam, Leeftijd) VALUES ('Jan',18);". The right window shows the implicit syntax: "INSERT INTO Tabel01 VALUES ('Jan',18);". Both scripts also include creating a database, creating a table, and selecting all from the table.

De leeg gelaten kolommen krijgen automatisch een waarde NULL toegekend. Deze NULL waarde is niet hetzelfde als het woord 'NULL' of het getal 0, maar betekent dat het veld niet ingevuld is en de waarde ervan onbekend is.

Op de tweede manier hoeft er minder ingetypt te worden, maar is het niet mogelijk om kolommen leeg te laten.

Voorbeeld:

```
INSERT INTO Tabel01 VALUES ('Mohammad',21);
SELECT * FROM Tabel01;
```



The screenshot shows two SSMS windows side-by-side. Both windows have the title 'SQLQuery1 - MON-ADIN001\Defenengen (SDF)\MSP02 (MSP)' - Microsoft SQL Server Management Studio. The left window contains the following SQL code:

```
USE Oefeningen;

CREATE TABLE Tabel01 (Naam VARCHAR(25), Leeftijd INT);

SELECT * FROM Tabel01;

INSERT INTO Tabel01 (Naam, Leeftijd) VALUES ('Jan',18);
INSERT INTO Tabel01 (Naam) VALUES ('Kees');
INSERT INTO Tabel01 (Leeftijd) VALUES (20);

SELECT * FROM Tabel01;

INSERT INTO Tabel01 VALUES ('Muhammad',21);
```

The right window contains the same SQL code:

```
CREATE TABLE Tabel01 (Naam VARCHAR(25), Leeftijd INT);

SELECT * FROM Tabel01;

INSERT INTO Tabel01 (Naam, Leeftijd) VALUES ('Jan',18);
INSERT INTO Tabel01 (Naam) VALUES ('Kees');
INSERT INTO Tabel01 (Leeftijd) VALUES (20);

SELECT * FROM Tabel01;

INSERT INTO Tabel01 VALUES ('Muhammad',21);

SELECT * FROM Tabel01;
```

Both windows show a successful execution message at the bottom: 'Query executed successfully.'

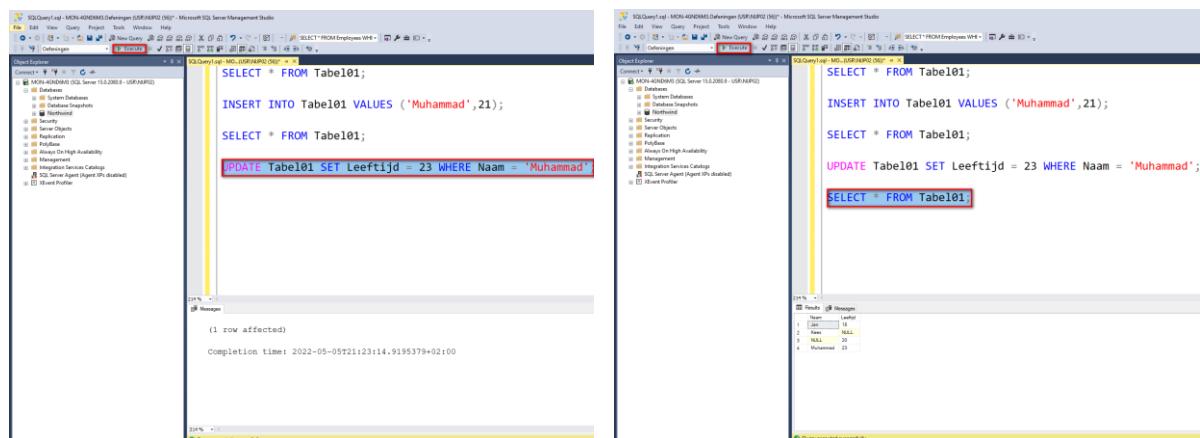
## 9.5 UPDATE TABLE

We kunnen gegevens in een tabel wijzigen met het UPDATE TABLE commando. De SQL syntax hiervoor is:

```
UPDATE <tablenaar> SET <kolomnaam> = <nieuwe waarde> WHERE <clausule>;
```

De WHERE clausule is hierbij gelijk aan die bij een SELECT statement.

```
UPDATE Tabel01 SET Leeftijd = 23 WHERE Naam = 'Mohammad';
SELECT * FROM Tabel01;
```



The screenshot shows two SSMS windows side-by-side. Both windows have the title 'SQLQuery1 - MON-ADIN001\Defenengen (SDF)\MSP02 (MSP)' - Microsoft SQL Server Management Studio. The left window contains the following SQL code:

```
SELECT * FROM Tabel01;

INSERT INTO Tabel01 VALUES ('Muhammad',21);

SELECT * FROM Tabel01;

UPDATE Tabel01 SET Leeftijd = 23 WHERE Naam = 'Muhammad';
```

The right window contains the same SQL code:

```
SELECT * FROM Tabel01;

INSERT INTO Tabel01 VALUES ('Muhammad',21);

SELECT * FROM Tabel01;

UPDATE Tabel01 SET Leeftijd = 23 WHERE Naam = 'Muhammad';

SELECT * FROM Tabel01;
```

Both windows show a successful execution message at the bottom: 'Query executed successfully.'

Wanneer er geen WHERE clausule wordt toegevoegd dan wordt de kolom voor alle rijen aangepast.

Voorbeeld:

```
UPDATE Tabel01 SET Leeftijd = 23;
SELECT * FROM Tabel01;
```

```
SQLQuery1 - MON-ADMIN01\Defeniger (SPP-NP02 DM) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Windows Help
SELECT * FROM Tabel01;
INSERT INTO Tabel01 VALUES ('Muhammad',21);
SELECT * FROM Tabel01;
UPDATE Tabel01 SET Leeftijd = 23 WHERE Naam = 'Muhammad';
SELECT * FROM Tabel01;
UPDATE Tabel01 SET Leeftijd = 23;
```

```
SQLQuery1 - MON-ADMIN01\Defeniger (SPP-NP02 DM) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Windows Help
SELECT * FROM Tabel01;
INSERT INTO Tabel01 VALUES ('Muhammad',21);
SELECT * FROM Tabel01;
UPDATE Tabel01 SET Leeftijd = 23 WHERE Naam = 'Muhammad';
SELECT * FROM Tabel01;
UPDATE Tabel01 SET Leeftijd = 23;
SELECT * FROM Tabel01;
```

## 9.6 ALTER TABLE

Wanneer we de structuur van een tabel willen aanpassen, zoals het toevoegen van een kolom, kunnen we dat doen met het ALTER TABLE commando. De SQL syntax daarvoor is:

```
ALTER TABLE <tabelnaam> ADD <kolomnaam> <datatype>;
```

Voorbeeld:

```
ALTER TABLE Tabel01 ADD Woonplaats VARCHAR(20);
SELECT * FROM Tabel01;
```

```
SQLQuery1 - MON-ADMIN01\Defeniger (SPP-NP02 DM) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Windows Help
SELECT * FROM Tabel01;
UPDATE Tabel01 SET Leeftijd = 23 WHERE Naam = 'Muhammad';
SELECT * FROM Tabel01;
UPDATE Tabel01 SET Leeftijd = 23;
SELECT * FROM Tabel01;
ALTER TABLE Tabel01 ADD Woonplaats VARCHAR(20);
```

```
SQLQuery1 - MON-ADMIN01\Defeniger (SPP-NP02 DM) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Windows Help
SELECT * FROM Tabel01;
UPDATE Tabel01 SET Leeftijd = 23 WHERE Naam = 'Muhammad';
SELECT * FROM Tabel01;
UPDATE Tabel01 SET Leeftijd = 23;
SELECT * FROM Tabel01;
ALTER TABLE Tabel01 ADD Woonplaats VARCHAR(20);
SELECT * FROM Tabel01;
```

De tabel heeft nu de nieuwe kolom Woonplaats, welke volledig gevuld is met NULL waarden.

## 9.7 DELETE FROM

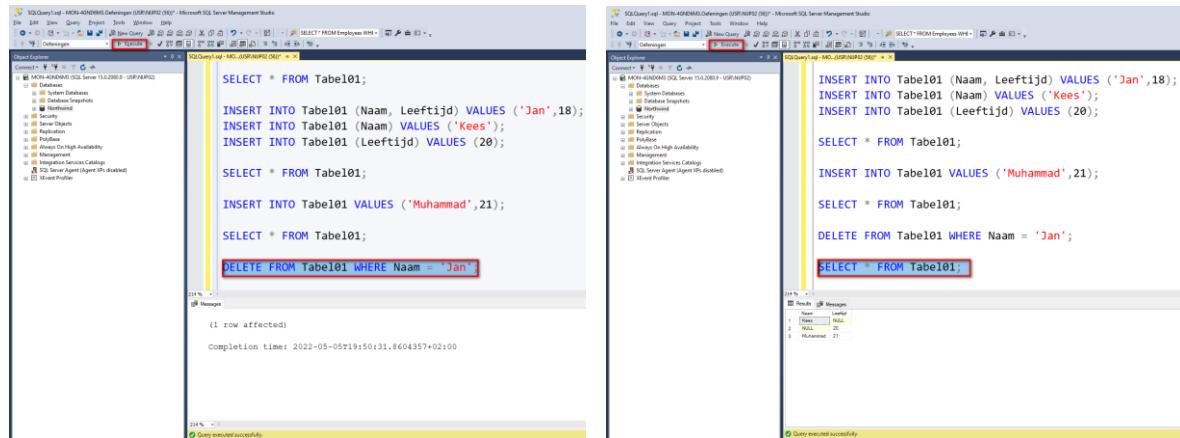
We kunnen rijen uit een tabel verwijderen met het DELETE commando. De SQL syntax hiervoor is:

```
DELETE FROM <tabelnaam> WHERE <clausule>;
```

De WHERE conditie is hetzelfde als bij SELECT statements wordt gebruikt.

Voorbeeld:

**DELETE FROM Tabel01 WHERE Naam = 'Jan';**



```

-- Query 1 (Left)
SELECT * FROM Tabel01;
INSERT INTO Tabel01 (Naam, Leeftijd) VALUES ('Jan', 18);
INSERT INTO Tabel01 (Naam) VALUES ('Kees');
INSERT INTO Tabel01 (Leeftijd) VALUES (20);

SELECT * FROM Tabel01;

INSERT INTO Tabel01 VALUES ('Muhammad', 21);

SELECT * FROM Tabel01;

DELETE FROM Tabel01 WHERE Naam = 'Jan';

-- Result of Query 1
(1 row affected)

-- Query 2 (Right)
SELECT * FROM Tabel01;
INSERT INTO Tabel01 (Naam, Leeftijd) VALUES ('Jan', 18);
INSERT INTO Tabel01 (Naam) VALUES ('Kees');
INSERT INTO Tabel01 (Leeftijd) VALUES (20);

SELECT * FROM Tabel01;

INSERT INTO Tabel01 VALUES ('Muhammad', 21);

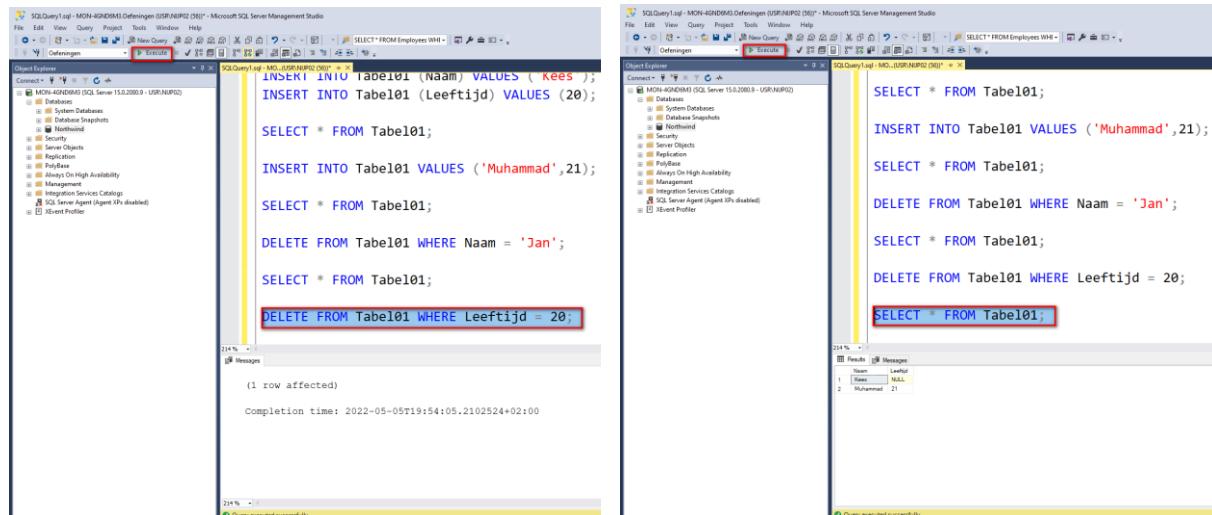
SELECT * FROM Tabel01;

DELETE FROM Tabel01 WHERE Naam = 'Jan';

SELECT * FROM Tabel01;

-- Result of Query 2
Name    Leeftijd
---    -----
Kees      NULL
Muhammad 21
  
```

**DELETE FROM Tabel01 WHERE Leeftijd = 20;**



```

-- Query 1 (Left)
SELECT * FROM Tabel01;
INSERT INTO Tabel01 (Naam) VALUES ('Kees');
INSERT INTO Tabel01 (Leeftijd) VALUES (20);

SELECT * FROM Tabel01;

INSERT INTO Tabel01 VALUES ('Muhammad', 21);

SELECT * FROM Tabel01;

DELETE FROM Tabel01 WHERE Naam = 'Jan';

SELECT * FROM Tabel01;

DELETE FROM Tabel01 WHERE Leeftijd = 20;

-- Result of Query 1
(1 row affected)

-- Query 2 (Right)
SELECT * FROM Tabel01;
INSERT INTO Tabel01 VALUES ('Muhammad', 21);

SELECT * FROM Tabel01;

DELETE FROM Tabel01 WHERE Naam = 'Jan';

SELECT * FROM Tabel01;

DELETE FROM Tabel01 WHERE Leeftijd = 20;

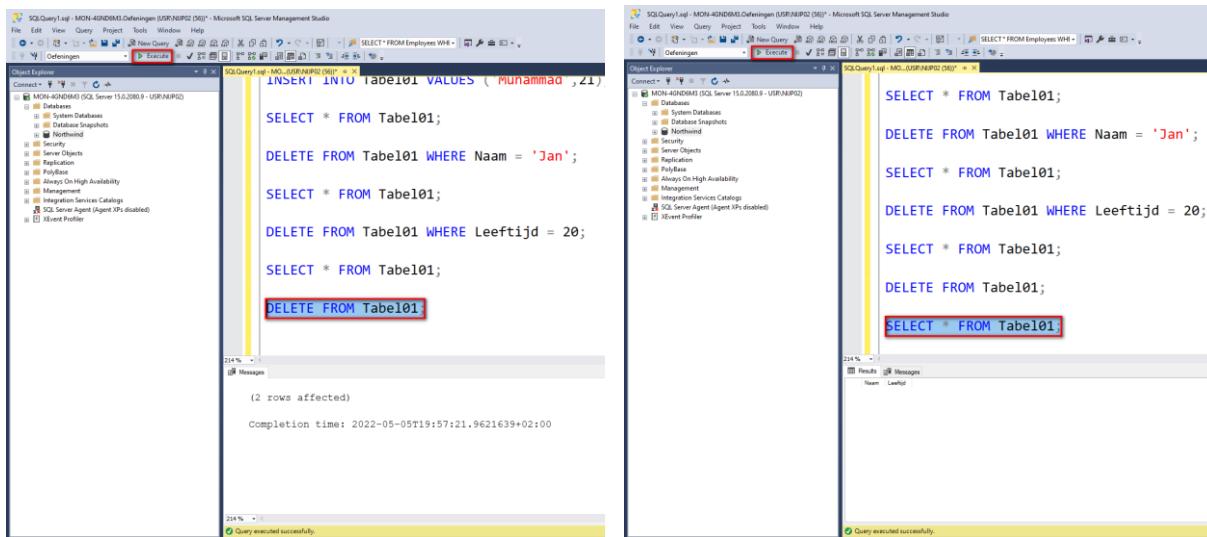
SELECT * FROM Tabel01;

-- Result of Query 2
Name    Leeftijd
---    -----
Muhammad 21
  
```

Wanneer geen WHERE clausule wordt gebruikt dan worden alle gegevens in de tabel verwijderd.

Voorbeeld:

**DELETE FROM Tabel01;**



The screenshot shows two SSMS windows. The left window contains the following SQL code:

```

CREATE TABLE Tabel01 (Naam VARCHAR(25), Leeftijd INT);
SELECT * FROM Tabel01;

INSERT INTO Tabel01 (Naam, Leeftijd) VALUES ('Jan',18);
DELETE FROM Tabel01 WHERE Naam = 'Jan';

SELECT * FROM Tabel01;
DELETE FROM Tabel01 WHERE Leeftijd = 20;

SELECT * FROM Tabel01;
DELETE FROM Tabel01;

```

The right window contains the following SQL code:

```

SELECT * FROM Tabel01;
DELETE FROM Tabel01 WHERE Naam = 'Jan';
SELECT * FROM Tabel01;
DELETE FROM Tabel01 WHERE Leeftijd = 20;
SELECT * FROM Tabel01;
DELETE FROM Tabel01;
SELECT * FROM Tabel01;

```

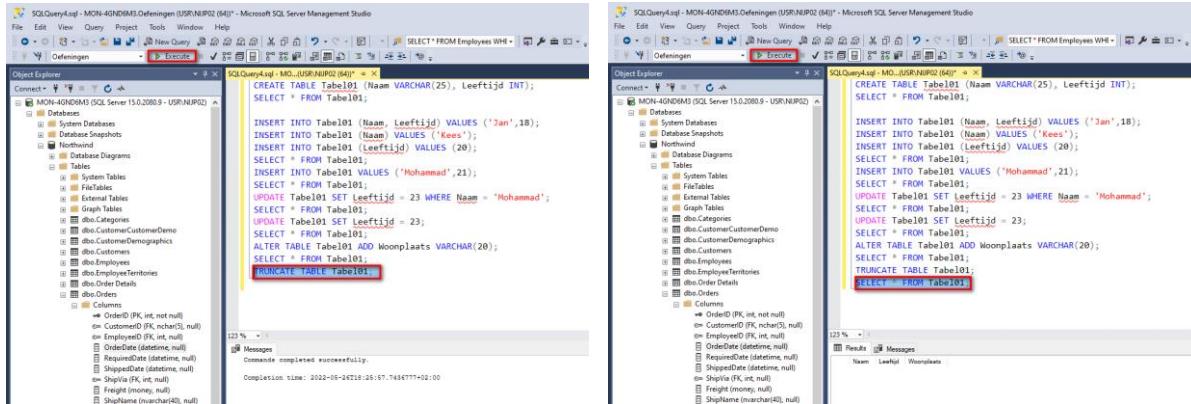
## 9.8 TRUNCATE TABLE

Een alternatieve manier om alle gegevens uit een tabel te verwijderen is d.m.v. het TRUNCATE commando.

Het TRUNCATE commando betreft net zoals de CREATE- en DROP commando's een DDL commando. Het TRUNCATE commando verwijdert in één keer alle rijen uit de tabel, terwijl een DELETE commando rij voor rij uit de tabel verwijdert. Een TRUNCATE is dus aanzienlijk sneller dan een DELETE, echter kan deze actie binnen een transactie niet ongedaan worden gemaakt omdat TRUNCATE een DDL commando is en geen DML commando. Transacties zullen in hoofdstuk 26 worden behandeld.

Voorbeeld:

**TRUNCATE TABLE Tabel01;**  
**SELECT \* FROM Tabel01;**



The screenshot shows two SSMS windows. The left window contains the following SQL code:

```

CREATE TABLE Tabel01 (Naam VARCHAR(25), Leeftijd INT);
SELECT * FROM Tabel01;

INSERT INTO Tabel01 (Naam, Leeftijd) VALUES ('Jan',18);
INSERT INTO Tabel01 (Naam) VALUES ('Kees');
INSERT INTO Tabel01 (Leeftijd) VALUES (20);
SELECT * FROM Tabel01;
INSERT INTO Tabel01 VALUES ('Mohammad',21);
SELECT * FROM Tabel01;
UPDATE Tabel01 SET Leeftijd = 23 WHERE Naam = 'Mohammad';
SELECT * FROM Tabel01;
UPDATE Tabel01 SET Leeftijd = 23;
SELECT * FROM Tabel01;
ALTER TABLE Tabel01 ADD Woonplaats VARCHAR(20);
SELECT * FROM Tabel01;
TRUNCATE TABLE Tabel01;

```

The right window contains the following SQL code:

```

CREATE TABLE Tabel01 (Naam VARCHAR(25), Leeftijd INT);
SELECT * FROM Tabel01;

INSERT INTO Tabel01 (Naam, Leeftijd) VALUES ('Jan',18);
INSERT INTO Tabel01 (Naam) VALUES ('Kees');
INSERT INTO Tabel01 (Leeftijd) VALUES (20);
SELECT * FROM Tabel01;
INSERT INTO Tabel01 VALUES ('Mohammad',21);
SELECT * FROM Tabel01;
UPDATE Tabel01 SET Leeftijd = 23 WHERE Naam = 'Mohammad';
SELECT * FROM Tabel01;
UPDATE Tabel01 SET Leeftijd = 23;
SELECT * FROM Tabel01;
ALTER TABLE Tabel01 ADD Woonplaats VARCHAR(20);
SELECT * FROM Tabel01;
TRUNCATE TABLE Tabel01;
SELECT * FROM Tabel01;

```

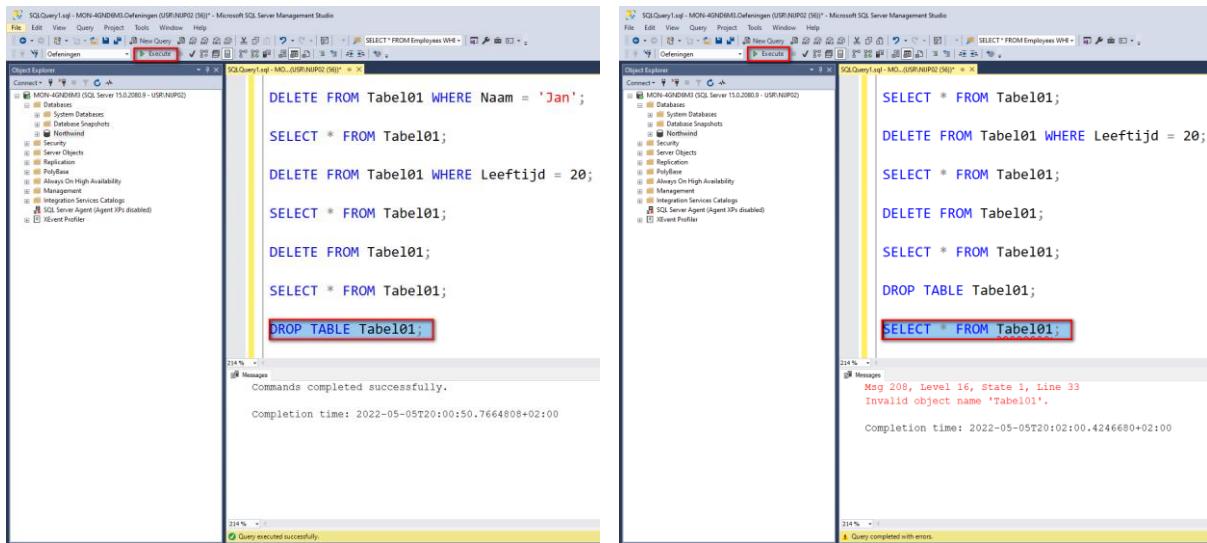
## 9.9 DROP TABLE

Een tabel kan worden verwijderd met het DROP commando. De SQL syntax hiervoor is:

**DROP TABLE <tabelnaam>;**

Voorbeeld:

```
DROP TABLE Tabel01;
SELECT * FROM Tabel01;
```



The screenshot shows two side-by-side SQL Server Management Studio windows.

**Left Window (Successful Creation):**

```

CREATE TABLE Tabel01
(
    Naam NVARCHAR(50),
    Leeftijd INT
);

INSERT INTO Tabel01
VALUES ('Jan', 20),
       ('Peter', 25),
       ('Sarah', 30),
       ('John', 35),
       ('Emily', 40);

SELECT * FROM Tabel01;

```

Execution results:

- Commands completed successfully.
- Completion time: 2022-05-05T20:00:50.7664808+02:00
- Query executed successfully.

**Right Window (Failed Drop):**

```

DROP TABLE Tabel01;

SELECT * FROM Tabel01;

```

Execution results:

- Msg 208, Level 16, State 1, Line 33  
Invalid object name 'Tabel01'.
- Completion time: 2022-05-05T20:02:00.4246680+02:00
- Query completed with errors.

## 9.10 DROP DATABASE

Een database kan worden verwijderd met het **DROP** commando. De SQL syntax hiervoor is:

```
DROP DATABASE <databasenaam>;
```

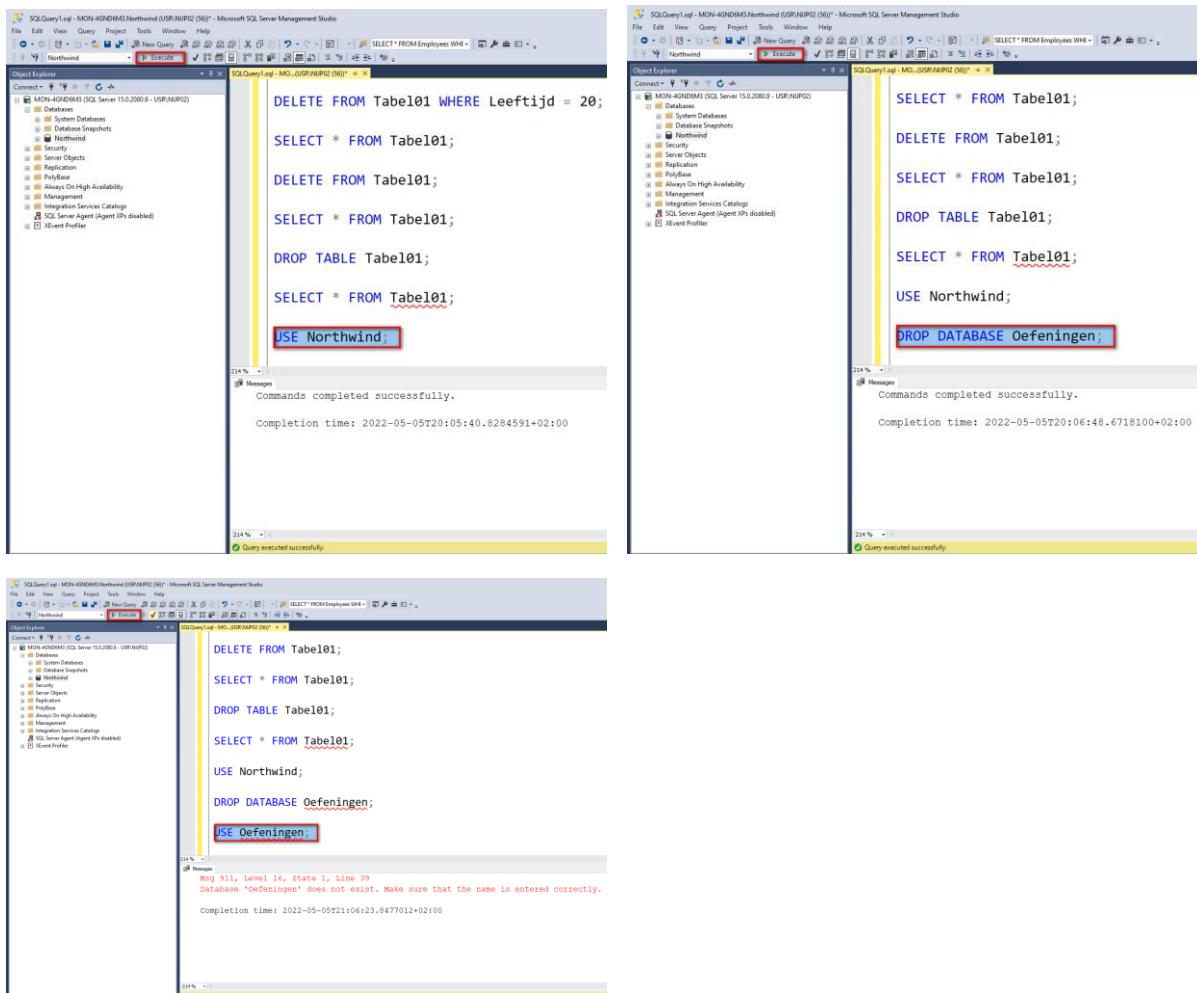
Het is echter niet mogelijk om een database te verwijderen wanneer er nog een gebruiker gebruik van maakt. Wanneer jij zelf de database courant hebt gemaakt - en dus de database in gebruik hebt - kan je daarom de database niet verwijderen en moet je eerst een andere database courant maken alvorens de database te kunnen verwijderen.

Voorbeeld:

```
USE Northwind;
```

```
DROP DATABASE Oefeningen;
```

```
USE Oefeningen;
```



```

-- Screenshot 1: Script to drop and create database 'Oefeningen'
DROP DATABASE Oefeningen;
CREATE DATABASE Oefeningen;

-- Screenshot 2: Script to create database 'Oefeningen' (fails)
CREATE DATABASE Oefeningen;

-- Screenshot 3: Script to drop and create database 'Oefeningen'
DROP DATABASE Oefeningen;
CREATE DATABASE Oefeningen;
    
```

## 9.11 SQL Scripts (NEW)

Voor het aanmaken of wijzigen van objecten wordt meestal gebruik gemaakt van SQL scripts. De instructies in deze SQL scripts worden sequentieel uitgevoerd (op volgorde zoals ze in het script staan). SQL scripts hebben een extensie .sql achter het bestand staan.

Het is gebruikelijk dat er commentaar in de SQL scripts wordt gezet om het script voor de lezer leesbaar te maken en om zo nodig uitleg te geven waarom bepaalde keuzes in het script zijn gemaakt.

Er zijn twee manier om commentaar in de scripts te zetten:

- **/\* <tekst> \*/** om meerdere regels in één keer om te zetten naar commentaar. Alles wat tussen **/\*** en **\*/** staat wordt als tekst gezien en wordt door SQL niet als SQL code uitgevoerd.
- **-- <commentaarregel>** om één regel om te zetten naar commentaar. Alles wat achter **--** staat wordt als tekst gezien en wordt door SQL niet als SQL code uitgevoerd. De volgende regel wordt wel weer als SQL code gezien.

We kunnen commentaar in de output van het SQL script zetten (rapport of logfile) d.m.v.:

**PRINT <tekst die in output terecht moet komen>**

ROC MONDRIAAN

## Voorbeeld van SQL-script

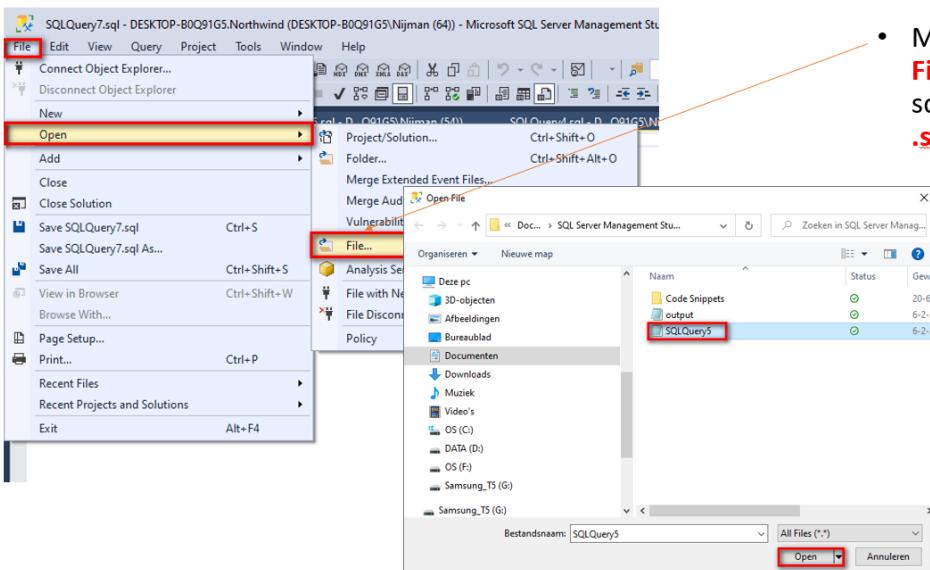
- Commentaar als documentatie van SQL-script tussen /\* en \*/
  - Commentaar als documentatie in rapport m.b.v. **PRINT**
  - Commentaar als documentatie na --
  - SQL-query
  - Commentaar als documentatie in rapport m.b.v. **PRINT**

6

We kunnen SQL scripts in SSMS openen m.b.v. File -> Open -> File

roc MONDRIAAN

# Openen van SQL-script

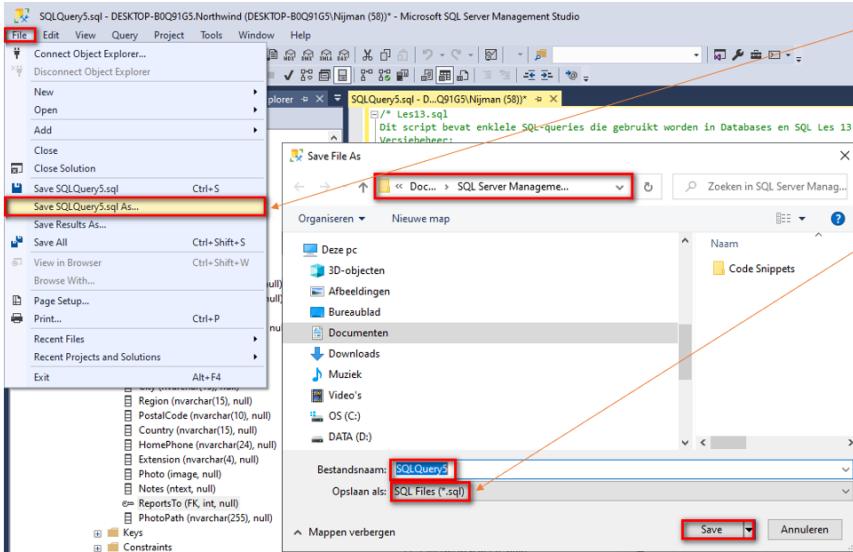


- Met **File -> Open - File** kan een SQL-script (met extensie **.sql**) worden geopend

We kunnen een SQL script opslaan in een bestand d.m.v. File -> Save SQLQuery AS ...

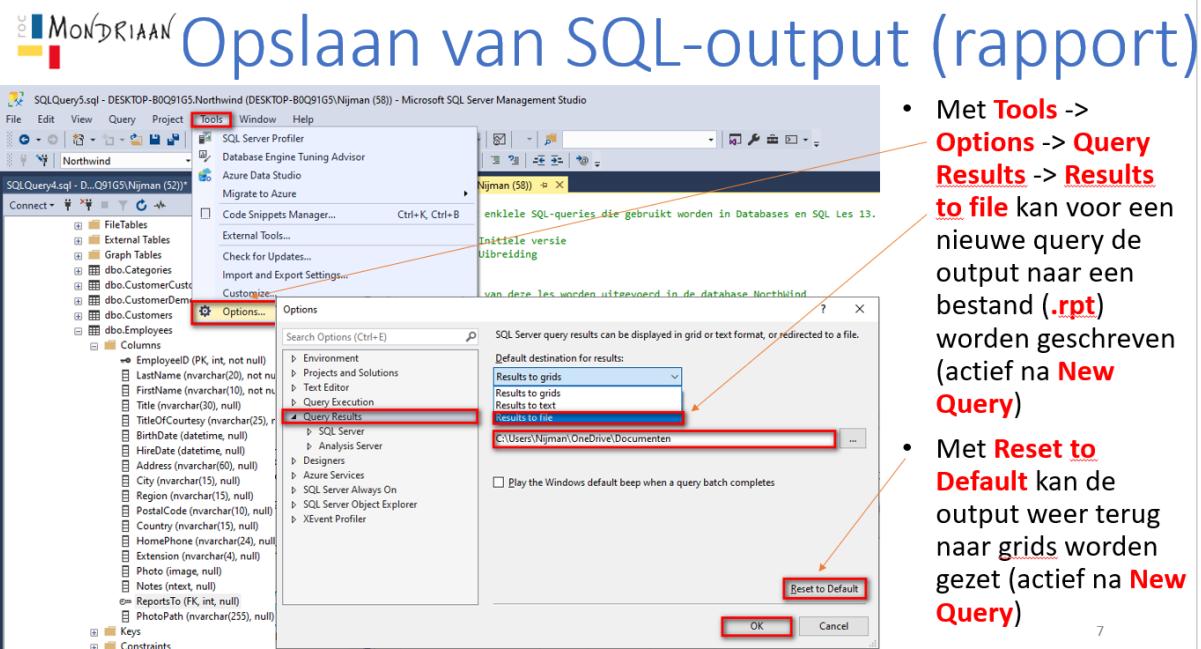


## Opslaan van SQL-script



- Met **File -> Save SQLQuery.sql As...**  
kan een script met  
SQL statements naar  
een bestand worden  
opgeslagen
- Het bestand krijgt de  
**extensie .sql**

We kunnen de output van een SQL script naar een bestand schrijven m.b.v. Tools -> Options -> Query Results -> Results to file.:



- Met **Tools -> Options -> Query Results -> Results to file** kan voor een nieuwe query de output naar een bestand (**.rpt**) worden geschreven (actief na **New Query**)
- Met **Reset to Default** kan de output weer terug naar **grids** worden gezet (actief na **New Query**)

De output van het SQL script wordt geschreven naar een output bestand (rapport of logfile) met de extensie **.rpt**, echter is dat voor de huidige sessie nog niet actief en moet er eerst een nieuwe sessie worden opgestart met **New Query**.



## Voorbeeld SQL-output (rapport)

TeBestellenProducten - Kladblok

Bestand Bewerken Opmaak Beeld Help

NorthWind Rapport: TE BESTELLEN PRODUCTEN

De volgende produkten hebben een voorraad minder dan 10:

ProductID	ProductName	SupplierID	UnitPrice	UnitsInStock	UnitsOnOrder
5	Chef Anton's Gumbo Mix	2	21,35	0	0
66	Louisiana Hot Spiced Okra	2	17,00	4	100
8	Northwoods Cranberry Sauce	3	46,00	6	0
74	Longlife Tofu	4	16,00	4	20
17	Alice Mutton	7	39,00	0	0
21	Sir Rodney's Scones	8	10,00	3	40
68	Scottish Longbreads	8	12,50	6	10
29	Thüringer Rostbratwurst	12	123,79	0	0
31	Gorgonzola Telino	14	12,50	0	70
32	Mascarpone Fabioli	14	32,00	9	40
45	Rogede sild	21	9,50	5	70
53	Perth Pasties	24	32,80	0	0

(12 rows affected)

Dit rapport is gegenereerd door Peter Nijman op 14 februari 2022

Completion time: 2022-02-13T22:48:31.7508768+01:00

Ln 1, Col 1 100% Windows (CRLF) UTF-8 met BOM

- Documentatie gemaakt door commando **PRINT**
- Output (resultaat) van SQL-query
- Documentatie gemaakt door commando **PRINT**

## Hoofdstuk 10 NOT NULL, UNIQUE, PRIMARY KEY Constraints

Tot nu toe hebben we steeds naar tabellen gekeken waarbij we alle waarden in alle velden zouden kunnen invullen. Niets hield ons tegen om onjuiste gegevens in te voeren (zoals studenten zonder studentnummer of studenten met hetzelfde studentnummer).

In deze en de volgende hoofdstukken gaan we kijken naar Constraints (beperkingen) die we op tabellen kunnen leggen zodat de gegevens die we in de tabel willen invoeren aan eisen moeten voldoen. Wanneer we een rij willen toevoegen waarvan een waarde niet aan de eisen voldoet dan wordt de gehele actie geweigerd en komt er een foutmelding.

### 10.1 Integriteitsregels

Enkele termen voor Integriteitsregels zijn:

- **Integriteitsregels** ofwel **integrity rules** ofwel **constraints** (beperkingen) vormen de regels waaraan de inhoud van een database te alle tijde behoort te voldoen en zij beschrijven daarmee welke mutaties op de database zijn toegestaan.
- **Integriteit** van gegevens is de consistentie en correctheid van gegevens. De integriteit geeft aan dat de gegevens in de database betrouwbaar zijn.
- **Consistentie** van gegevens betekent dat de gegevens elkaar niet tegenspreken. Een voorbeeld van elkaar tegensprekende gegevens zou bijvoorbeeld zijn als in de database staat dat studentnummer 129 is van de student Jan Jansen maar elders in de database staat dat studentnummer 129 is van de student Piet Hermans.
- **Correctheid** betekent dat de gegevens voldoen aan relevante (bedrijf)regels, natuurwetten etc. De gegevens moeten voldoen aan de werkelijkheid. Zo kan een medewerker bijvoorbeeld niet op twee afdelingen tegelijk werken daar dat volgens de bedrijfsregels niet kan, of kan een klant niet een leeftijd van 213 jaar hebben daar dat volgens de natuurwetten niet kan.

M.b.v. de constraints (beperkingen) kunnen we dus voorkomen dat waarden ingevoerd kunnen worden die niet voldoen aan de integriteitsregels. In deze en de volgende hoofdstukken zullen we ons bezig houden met de volgende integriteitsregels:

- **NOT NULL constraint** om te voorkomen dat er lege velden kunnen worden ingevoerd.
- **Unique constraint** om te voorkomen dat er dubbele gegevens kunnen worden ingevoerd.
- **Primary Key constraint , Candidate key constraint** en **Alternate key constraint** om af te dwingen dat elke rij in een tabel uniek geïdentificeerd is.
- **Foreign Key constraint** om af te dwingen dat een waarde in een kolom slechts kan worden ingevoerd indien deze waarde reeds voorkomt in een andere gerefereerde tabel.
- **Check constraint** om andere beperkingen voor (bedrijfs)regels af te dwingen.

## 10.2 NOT NULL Constraint

Met een NOT NULL Constraint kunnen we voorkomen dat bij het invoeren of aanpassen van gegevens een veld leeg (zonder waarde) wordt gelaten.

Stel dat we de volgende SQL statements uitvoeren:

## CREATE DATABASE OEFENINGEN;

## USE OEFENINGEN;

```
CREATE TABLE Tabel01 (Voornaam varchar(20), Achternaam varchar(20));
```

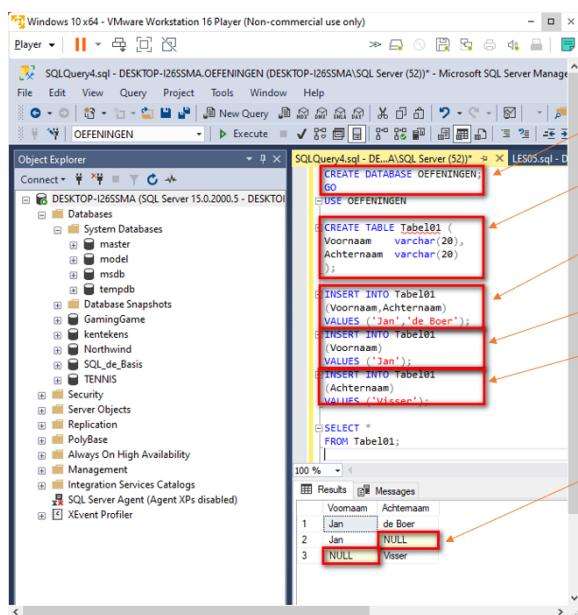
```
INSERT INTO Tabel01 (Voornaam,Achternaam) VALUES ('Jan','de Boer');
```

```
INSERT INTO Tabel01 (Voornaam) VALUES ('Jan');
```

**INSERT INTO Tabel01 (Achternaam) VALUES ('Visser');**

**SELECT \* FROM Tabel01;**

Dan is het resultaat als volgt:



- Creëren van database OEFENINGEN
  - Creëren van tabel Tabel01 met kolommen Voornaam en Achternaam
  - Rij invoeren in Tabel01 met Voornaam = Jan en Achternaam = de Boer
  - Invoeren van rij zonder achternaam in tabel01
  - Invoeren van rij zonder voornaam in tabel01
  - Merk op dat er daadwerkelijk twee lege velden zijn ingevoerd in Tabel01, die de waarde **NULL** hebben gekregen  
Dit kan **inconsistentie** in de database veroorzaken en is dan ongewenst!

We kunnen nu een NOT NULL constraints toevoegen door NOT NULL achter de kolomdefinities zetten:

```
CREATE TABLE Tabel02 (Voornaam varchar(20) NOT NULL, Achternaam varchar(20) NOT NULL);
```

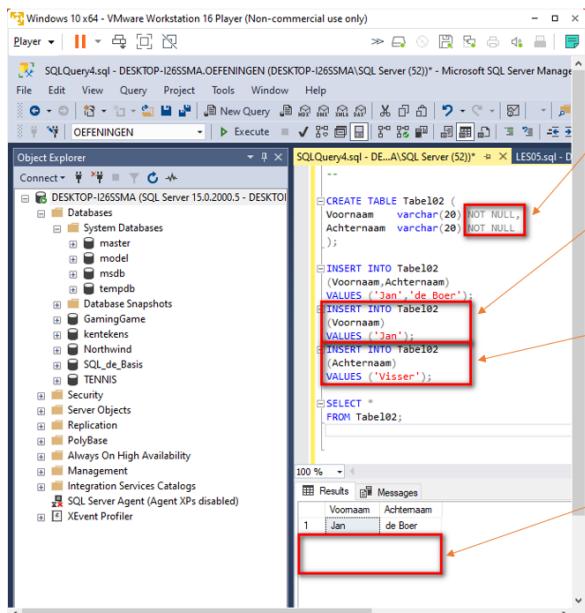
```
INSERT INTO Tabel02 (Voornaam,Achternaam) VALUES ('Jan','de Boer');
```

**INSERT INTO Tabel02 (Voornaam) VALUES ('Jan');**

**INSERT INTO Tabel02 (Achternaam) VALUES ('Visser');**

**SELECT \* FROM Tabel02;**

Het resultaat ervan is dan:



```

CREATE TABLE Tabel02 (
    Voornaam varchar(20) NOT NULL,
    Achternaam varchar(20) NOT NULL
)

INSERT INTO Tabel02
(Voornaam,Achternaam)
VALUES ('Jan','de Boer');

INSERT INTO Tabel02
(Voornaam)
VALUES ('Jan');

INSERT INTO Tabel02
(Achternaam)
VALUES ('Visser');

SELECT *
FROM Tabel02;

```

- De **NOT NULL** constraints op de kolommen Voornaam en Achternaam voorkomen dat er een veld leeg gelaten kan worden
- Invoeren van een rij zonder Achternaam geeft foutmelding:  
**Msg 515, Level 16, State 2, Line 12**  
*Cannot insert the value NULL into column 'Achternaam', table 'OEFENINGEN.dbo.Tabel02'; column does not allow nulls. INSERT fails.*
- Invoeren van een rij zonder Voornaam geeft foutmelding:  
**Msg 515, Level 16, State 2, Line 13**  
*Cannot insert the value NULL into column 'Voornaam', table 'OEFENINGEN.dbo.Tabel02'; column does not allow nulls. INSERT fails.*
- Merk op dat alleen de invoer met een voornaam en een achternaam is ingevoerd in Tabel02 en dat de rijen zonder voornaam of achternaam zijn geweigerd:  
**The statement has been terminated.**

Dus als we willen verplichten dat een kolom altijd ingevuld wordt, dan kunnen we een NOT NULL constraint toevoegen door NOT NULL achter de kolomdefinitie te zetten.

### 10.3 UNIQUE KEY Constraint

Met een UNIQUE KEY constraint kunnen we voorkomen dat er dubbele gegevens ingevoerd worden, ofwel ze kunnen ermee zorgen dat elke waarde in de kolom (of combinatie van kolommen) altijd uniek is.

Stel dat we de volgende SQL statements uitvoeren:

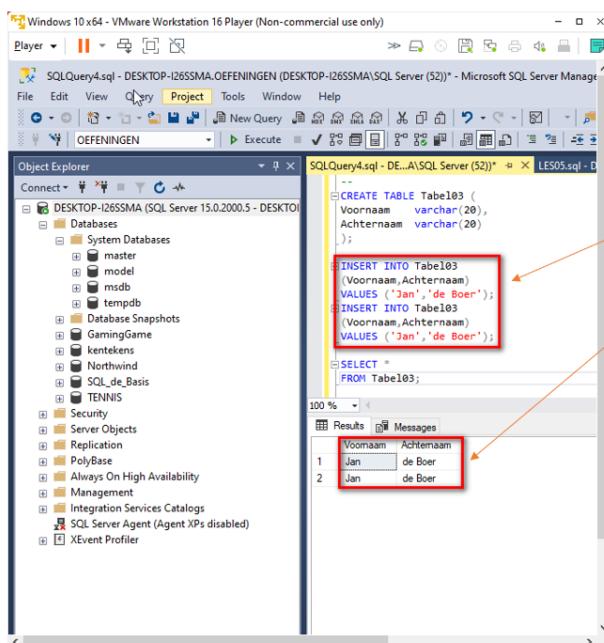
**CREATE TABLE Tabel03 (Voornaam varchar(20), Achternaam varchar(20));**

**INSERT INTO Tabel03 (Voornaam,Achternaam) VALUES ('Jan','de Boer');**

**INSERT INTO Tabel03 (Voornaam,Achternaam) VALUES ('Jan','de Boer');**

**SELECT \* FROM Tabel03;**

Het resultaat is dan:



The screenshot shows the Object Explorer and a query window. The query window contains the following code:

```

CREATE TABLE Tabel03 (
    Voornaam varchar(20),
    Achternaam varchar(20)
);

INSERT INTO Tabel03
(Voornaam,Achternaam)
VALUES ('Jan','de Boer');
INSERT INTO Tabel03
(Voornaam,Achternaam)
VALUES ('Jan','de Boer');

SELECT *
FROM Tabel03;

```

The results show two rows with the same values:

Voornaam	Achternaam
Jan	de Boer
Jan	de Boer

- Hier wordt twee maal dezelfde rij gegevens ingevoerd met Voornaam = Jan en Achternaam = de Boer
- Merk op dat er inderdaad twee maal dezelfde gegevens in de tabel voorkomt. Dit veroorzaakt **redundantie** (overbodige gegevens) in de database omdat de gegevens dubbel zijn. Redundante gegevens nemen onnodige (schijf)ruimte in gebruik en kunnen leiden tot **inconsistente** (tegenstrijdige) gegevens, en dat is ongewenst!

We kunnen een UNIQUE KEY constraint op een tabel toevoegen door:

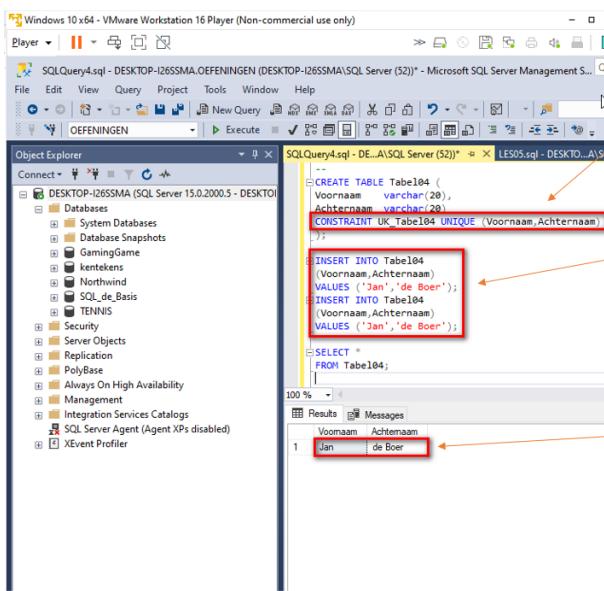
**CREATE TABLE Tabel04 (Voornaam varchar(20), Achternaam varchar(20)  
CONSTRAINT UK\_Tabel04 UNIQUE (Voornaam, Achternaam));**

**INSERT INTO Tabel04 (Voornaam,Achternaam) VALUES ('Jan','de Boer');**

**INSERT INTO Tabel04 (Voornaam,Achternaam) VALUES ('Jan','de Boer');**

**SELECT \* FROM Tabel04;**

Het resultaat is dan:



The screenshot shows the Object Explorer and a query window. The query window contains the following code:

```

CREATE TABLE Tabel04 (
    Voornaam varchar(20),
    Achternaam varchar(20)
);

CONSTRAINT UK_Tabel04 UNIQUE (Voornaam,Achternaam);

INSERT INTO Tabel04
(Voornaam,Achternaam)
VALUES ('Jan','de Boer');
INSERT INTO Tabel04
(Voornaam,Achternaam)
VALUES ('Jan','de Boer');

SELECT *
FROM Tabel04;

```

The results show one row with the values Jan and de Boer.

- De **Unique Key constraint** op de kolommen Voornaam en Achternaam voorkomt dat dubbele gegevens met dezelfde Voornaam en Achternaam kan worden ingevoerd
- Bij twee maal dezelfde gegevens invoeren voor Jan de Boer zal de tweede invoer een foutmelding krijgen:  
`Msg 2627, Level 14, State 1, Line 19  
Violation of UNIQUE KEY constraint  
'UK_Tabel04'. Cannot insert duplicate key  
in object 'dbo.Tabel04'. The duplicate  
key value is (Jan, de Boer).`  
**The statement has been terminated.**
- Merk op dat de gegevens van Jan de Boer zijn slechts één maal ingevoerd. De tweede invoer door de database is geweigerd.

In bovenstaand voorbeeld hebben we de UNIQUE KEY constraint op de tabel toegevoegd als een Tabel constraint omdat we de CONSTRAINT UNIQUE definitie na de kolomdefinities hebben

gedefinieerd. Dit is noodzakelijk wanneer de constraint op de combinatie van meerdere kolommen staat. Wanneer een constraint op slechts één kolom staat dan is het ook mogelijk om een UNIQUE KEY constraint op die ene kolom te plaatsen als een kolom constraint direct achter de kolomdefinitie.

Een **Unique Key constraint** kan op 2 manieren worden gedefinieerd:

```
CREATE TABLE Tabel04 (
    Voornaam  varchar(20),
    Achternaam varchar(20)
) CONSTRAINT UK_Tabel04 UNIQUE (Voornaam, Achternaam);

INSERT INTO Tabel14 Values ('Jan','Jansen');
INSERT INTO Tabel14 Values ('Jan','Jansen');
```

1. Als **tabel constraint**

Dit wordt gebruikt wanneer de constraint over meerdere kolommen gaat (nu Voornaam + Achternaam). Hierdoor wordt gegarandeerd dat elke combinatie van Voornaam en Achternaam in de tabel uniek is

2. Als **kolom constraint**

Dit wordt gebruikt wanneer de constraint slechts één kolom betreft (nu Voornaam). Hierdoor wordt gegarandeerd dat elke Voornaam uniek is.

```
CREATE TABLE Tabel04 (
    Voornaam  varchar(20) UNIQUE,
    Achternaam varchar(20) );

INSERT INTO Tabel14 Values ('Jan','Jansen');
INSERT INTO Tabel14 Values ('Jan','Klaasen');
```

Het voordeel van een kolom constraint is dat het minder typewerk geeft en duidelijk is dat het slechts die ene kolom betreft. Nadeel van een kolom constraint is dat het bij een kolomdefinitie niet mogelijk is om de constraint een sprekende naam te geven en zal SQL Server een naam genereren die voor ons niets zegt. Een kolom constraint kan echter niet worden gebruikt wanneer de constraints over de combinatie van meerdere kolommen gaan en dan zal een tabel constraint moeten worden gebruikt.

## 10.4 PRIMARY KEY Constraint

Een PRIMARY KEY constraint is functioneel hetzelfde als een combinatie van een NOT NULL en een UNIQUE KEY constraint. Door een Primary Key Constraint is het niet mogelijk om in de kolommen van de Primary Key dubbele gegevens of lege waarden in te voeren. Alle waarden in de Primary Key kolommen zijn dus gegarandeerd uniek en hebben géén NULL waarde. Daardoor is de Primary key Constraint geschikt om elke rij in de tabel uniek te identificeren. In elke tabel kan slechts één Primary Key constraint worden gedefinieerd.

Functioneel is een PRIMARY KEY constraint hetzelfde als een combinatie van een NOT NULL constraint en een UNIQUE KEY constraint, maar technisch zijn er echter wel verschillen tussen een PRIMARY KEY constraint en een combinatie van een UNIQUE KEY constraint met een NOT NULL constraint. Zo kan je bij een PRIMARY KEY constraint bijvoorbeeld automatische nummering gebruiken.

De SQL syntax voor een Primary Key constraint is:

```
CREATE TABLE <tabelnaam> (kolomnaam1 datatype, kolomnaam2 datatype, ...)
CONSTRAINT <constraintnaam> PRIMARY KEY (kolomnaam, kolomnaam,...);
```

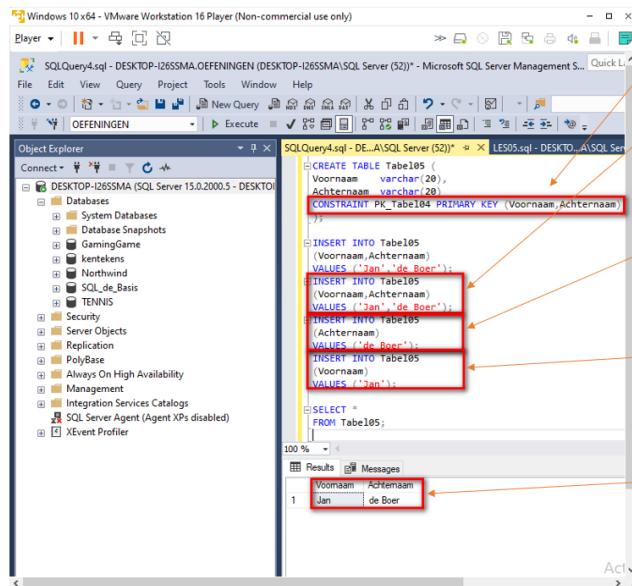
Als we bijvoorbeeld de volgende SQL statements uitvoeren:

```
CREATE TABLE Tabel09 (Voornaam varchar(20),Achternaam varchar(20)
CONSTRAINT PK_Tabel09 PRIMARY KEY (Voornaam, Achternaam));
```

```
INSERT INTO Tabel09 (Voornaam,Achternaam) VALUES ('Jan','de Boer');
INSERT INTO Tabel09 (Voornaam,Achternaam) VALUES ('Jan', 'de Boer');
INSERT INTO Tabel09 (Achternaam) VALUES ('de Boer');
INSERT INTO Tabel09 (Voornaam) VALUES ('Jan');
```

**SELECT \* FROM Tabel09;**

Dan is het resultaat:



The screenshot shows the SQL Server Management Studio interface. In the Object Explorer, a database named 'OEFENINGEN' is selected. In the center pane, a script window displays the following code:

```
CREATE TABLE Tabel09 (
    Voornaam varchar(20),
    Achternaam varchar(20)
CONSTRAINT PK_Tabel09 PRIMARY KEY (Voornaam,Achternaam)
);

INSERT INTO Tabel09
(Voornaam,Achternaam)
VALUES ('Jan','de Boer');
INSERT INTO Tabel09
(Voornaam,Achternaam)
VALUES ('Jan','de Boer');
INSERT INTO Tabel09
(Achternaam)
VALUES ('de Boer');
INSERT INTO Tabel09
(Voornaam)
VALUES ('Jan');

SELECT *
FROM Tabel09;
```

Below the code, the results pane shows a single row of data:

Voornaam	Achternaam
Jan	de Boer

- De Primary Key Constraint garandeert dat alle rijen uniek geïdentificeerd zijn en er geen lege velden zijn
- De dubbele invoer van Jan de Boer krijgt een foutmelding:  
Msg 2627, Level 14, State 1, Line 24  
Violation of PRIMARY KEY constraint  
'PK\_Tabel09'. Cannot insert duplicate key in  
object 'dbo.Tabel09'. The duplicate key value  
is (Jan, de Boer).
- Het weglaten van de Voornaam krijgt een foutmelding:  
Msg 515, Level 16, State 2, Line 25 Cannot  
insert the value NULL into column 'Voornaam',  
table 'OEFENINGEN.dbo.Tabel09'; column does  
not allow nulls. INSERT fails.
- Het weglaten van een Achternaam krijgt een foutmelding:  
Msg 515, Level 16, State 2, Line 26 Cannot  
insert the value NULL into column 'Achternaam',  
table 'OEFENINGEN.dbo.Tabel09'; column does not  
allow nulls. INSERT fails.
- De dubbele en ontbrekende gegevens zijn geweigerd en zitten niet in de tabel, melding:  
The statement has been terminated.

Het is echter onhandig om Voornaam en Achternaam als Primary Key constraint te gebruiken. Er kunnen immers personen bestaan met dezelfde naam en een vrouw kan na huwelijk de achternaam van haar man aannemen:

```
CREATE TABLE Tabel09 (Voornaam varchar(20), Achternaam varchar(20)
CONSTRAINT PK_Tabel09 PRIMARY KEY (Voornaam, Achternaam));
```

Het is daarom gebruikelijk om een numerieke kolom als Primary Key te gebruiken voor de tabel, bijvoorbeeld Studentno:

```
CREATE TABLE Tabel09 (Studentno int NOT NULL, Voornaam varchar(20) NOT NULL, Achternaam
varchar(20) NOT NULL, Burgerservicenummer varchar(9) NOT NULL UNIQUE,
CONSTRAINT PK_Tabel09 PRIMARY KEY (Studentno));
```

Net als de Unique key constraint kan de Primary key constraint worden gedefinieerd als tabel constraint of als kolom constraint wanneer de PRIMARY KEY constraint slechts één kolom bevat:

```
CREATE TABLE Tabel09 (Studentno int NOT NULL PRIMARY KEY, Voornaam varchar(20) NOT NULL,
Achternaam varchar(20) NOT NULL, Burgerservicenummer varchar(9) NOT NULL UNIQUE);
```

Wanneer een kolom van een Primary Key constraint vooraf niet als NOT NULL is gedefinieerd dan zal dat alsnog door de Primary Key constraint impliciet op NOT NULL worden gezet. Maar het is voor de duidelijkheid beter om de NOT NULL constraint reeds expliciet op elke kolom van de Primary key te definieren.

## 10.5 ALTER TABLE ... ADD PRIMARY KEY Constraint

Naast het creeren van een PRIMARY KEY constraint in een CREATE TABLE statement kunnen we ook een PRIMARY KEY constraint toevoegen nadat een tabel reeds is aangemaakt.

De SQL syntax hiervoor is:

```
ALTER TABLE <tabelnaam> ADD PRIMARY KEY (<kolommen>);
```

## 10.6 CANDIDATE KEY Constraint en ALTERNATE KEY Constraint

In een tabel kunnen meerdere kolommen of kolomcombinaties aanwezig zijn die de rijen in de tabel uniek identificeren, zoals Studentnummer, BurgerServiceNummer (BSN), Paspoortnummer, Rijbewijsnummer, etc. Het zijn allemaal nummers die uniek zijn en niet leeg kunnen zijn. Er zijn immers géén personen met eenzelfde BSN-nummer en er zijn geen personen zonder BSN-nummer.

Alle unieke kolommen (of kolomcombinaties) in een tabel noemen we de CANDIDATE KEY constraints, ofwel zijn allen kandidaat om de PRIMARY KEY constraint te worden. In elke tabel kan echter slechts één PRIMARY KEY constraint worden gedefinieerd. Één geselecteerde CANDIDATE KEY constraint wordt gekozen tot PRIMARY KEY constraint, de overige CANDIDATE KEY constraints worden dan ALTERNATE KEY constraints genoemd.

Dus CANDIDATE KEYS = PRIMARY KEY + ALTERNATE KEYS.

## Hoofdstuk 11 FOREIGN KEY Constraints

In paragraaf 1.2 hadden we geleerd dat we een grote tabel met meerder gegevensgroepen Studentgegevens en Klasgegevens konden opdelen in kleinere tabellen met slechts één gegevensgroep per tabel, bijvoorbeeld de tabel Studenten en de tabel Klassen.

A	B	C	D	E	F	G	H	I	J	K
1 Studentnummer	Voorletters	Achternaam	Geslacht	geboortedatum	Adres	Huisnummer	Postcode	Woonplaats	telefoonnummer	Klas
2 S0001	A.	Baan	M	12-5-2004	Terwestenstraat	18	2525GH	Den Haag	06-12345678	1A
3 S0002	S.	Hijmans	V	7-12-2004	Sweelinckplein	16	2517GM	Den Haag	06-23456789	1B
4 S0003	J.	Jansen	M	14-3-2004	Brinkershof	17	2713TX	Zoetermeer	071-1234567	1C
5 S0004	J.	Jansen	M	14-3-2004	Brinkershof	17	2713TX	Zoetermeer	071-1234567	1C
6 S0005	A.	Maarsen	M	12-5-2004	Rembrandtstraat	2	2526PZ	Den Haag	06-56789012	1D
7 S0006	K.	Jakobs	V	12-12-2005	De Vliegerstraat	55	2525BG	Den Haag	06-67890123	1B
8 S0007	A.B.	Willemsen	M	14-5-2004	Terwestenstraat	16	2525GH	Den Haag	06-78901234	1B
9 S0008	U.	Jans	M	17-6-2005	Herman Costerstraat	162	2571PD	Den Haag	06-89012345	1C
10 S0009	S.	Huber	M	12-5-2004	Hobbemastraat	22	2526P	Den Haag	06-90123456	1A
11 S0010	Z.	Wessels	V	5-8-2004	Rietveldstraat	29	2571PS	Den Haag	06-13579135	1B
12 S0011	A.	Kassenberg	M	12-5-2006	Almeloplein	12	2533AA	Den Haag	06-24680246	1D
13 S0012	L.	Vrogers	M	12-5-2004	van der Vennestraat	85	2525CC	Den Haag	06-35791357	1C
14 S0013	A.D.	Elfering	M	5-9-2005	Reitzstraat	52	2571RT	Den Haag	06-57913579	1A
15 S0014	I.	Sebastiaans	M	12-5-2004	Rembrandtstraat	1	2526PN	Den Haag	06-79135791	1C
16 S0015	J.	de Jong	M	12-12-2006	van Miereveldstraat	62	2525EG	Den Haag	06-91357913	1D
17 S0016	T.	Boeren	V	12-8-2004	Herman Costerstraat	164	2571PD	Den Haag	06-13579135	1B
18 S0017	J.	Huizinga	M	12-5-2005	Terwestenstraat	20	2525GH	Den Haag	06-25678901	1D
19 S0018	D.	Pol	M	12-12-2004	Wesselstraat	283	2572RZ	Den Haag		1A
20 S0019	E.	Reitsema	M	12-5-2004	Rietveldstraat	27	2571PS	Den Haag	06-47890123	1A
21 S0020	P.	Vos	M	5-3-2004	van Miereveldstraat	64	2525EG	Den Haag	06-69012345	1B
22 S0021	K.	Hazenberg	V	12-5-2004	van der Vennestraat	83	2525CC	Den Haag	06-71234567	1D
23 S0022	L.	Joemman	M	12-5-2005	Hobbemastraat	20	2526IP	Den Haag	06-94567890	1A
24 S0023	S.	Claasen	-	8-9-2004	Terwestenstraat	111	2525GG	Den Haag	06-10987654	1C
25 S0024	W.	de Wit	M	12-5-2004	Almeloplein	10	2533AA	Den Haag	06-29876543	1D
26 S0025	O.	Snellers	M	12-10-2005	Herman Costerstraat	160	2571PD	Den Haag	06-30986543	1A
27 H.	Blokhuis	V	8-10-2006	Rembrandtstraat	3	2526PN	Den Haag	06-41234098	1D	
28 S0026	D.	Haverkamp	V	12-5-2004	Rietveldstraat	25	2571PS	Den Haag	06-52345679	1B
29 S0026	B.	Bol	M	12-12-2006	Terwestenstraat	22	2525GH	Den Haag	06-59873456	1D
30 S0027	K.	Reuvenkamp	M	22-11-2005	Reitzstraat	50	2571RT	Den Haag	06-59873112	1D
31 S0028	L.	van den Broek	M	12-5-2004	Wesselstraat	281	2572RZ	Den Haag	06-22095586	1C

A	B	C	D	E	F
1 Klas	Locatie	Adres	Postcode	Plaats	Klassedocent
2 1A	Gebouw H	Wegastraat 56	2516CJ	Den Haag	A.P. Jansen
3 1B	Gebouw R	Volmerlaan 12	2288GD	Rijswijk	J. Mooiweer
4 1C	Gebouw Z	Houtsingel 91	2719EB	Zoetermeer	K. Damen
5 1D	Gebouw D	Oude Delft 12	2611OC	Delft	S. Slooten

Tussen deze twee tabellen bestaat een relatie op basis van de kolom Klas: Elke student zit in een bepaalde klas en elke klas zit op een bepaalde locatie. Er bestaat dus een relatie tussen de kolom Klas in de tabel Studenten en de kolom Klas in de tabel Klassen. Let wel: Er is géén directe relatie tussen een student en een locatie! Het is dus niet zo dat de studenten zijn ingedeeld op een bepaalde locatie, maar de studenten zijn ingedeeld in klassen en de klassen zijn ingedeeld over locaties. Deze relatie kunnen we in de database implementeren m.b.v. een Foreign Key constraint.

Met een Foreign Key constraint kunnen we ervoor zorgen dat in een kolom (of combinatie van kolommen) alleen een waarde ingevuld kan worden die in een andere tabel reeds bestaat. Het is bijvoorbeeld in bovenstaand voorbeeld niet mogelijk om in tabel Studenten een student in te voeren met een klas 1E omdat klas 1E nog niet in tabel Klassen bestaat. Mocht het zo zijn dat er daadwerkelijk een student in klas 1E blijkt te zitten, dan dient er eerst een rij in tabel Klassen te worden ingevoerd met de waarde Klas 1E. Pas daarna is het mogelijk om een student in te voeren met klas 1E in tabel Studenten.

Telkens wanneer men een waarde wil invullen dan controleert de Foreign Key constraint of de ingevulde waarde wel in de andere tabel aanwezig is. Als de waarde niet in de andere tabel aanwezig is dan zal de Foreign Key constraint de actie in zijn geheel weigeren en een foutmelding geven. In bovenstaand voorbeeld betekent dat telkens als we een student willen toevoegen in de tabel Studenten de Foreign Key constraint op de tabel Studenten eerst controleert in tabel Klassen of de waarde die we bij de kolom Klas invullen reeds in tabel Klassen staat. Als dat wel het geval is dan zal de Foreign Key constraint de actie toelaten maar als dat niet het geval is dan zal de Foreign Key constraint de gehele actie weigeren en een foutmelding geven.

### 11.1 FOREIGN KEY Constraint

FOREIGN KEY constraints (Refererende integriteitsregels) zijn dus relaties tussen tabellen onderling waarbij de waarde in de ene tabel moet voorkomen in de andere tabel.

De tabel waarnaar door de FOREIGN KEY constraint gerefereerd wordt, wordt de **gerefereerde tabel** genoemd en de tabel die naar de gerefereerde tabel verwijst wordt de **refererende tabel** genoemd.

We kunnen een Foreign Key constraint aanmaken op een reeds bestaande tabel of we kunnen een Foreign Key constraint aanmaken op het moment dat we de tabel zelf aanmaken. De gerefereerde tabel moet dus een reeds bestaande tabel zijn of moet de tabel zelf zijn die gecreëerd wordt. Dit laatste is het geval indien de refererende tabel dezelfde tabel is als de gerefereerde tabel (een tabel met een FOREIGN KEY constraint die verwijst naar zijn eigen PRIMARY KEY constraint).

## 11.2 ALTER TABLE ... ADD FOREIGN KEY Constraint

We kunnen een Foreign Key constraint aanmaken op een reeds bestaande tabel met de SQL syntax:

```
ALTER TABLE <tabelnaam> ADD CONSTRAINT <constraintnaam> FOREIGN KEY (<kolomnamen>)
REFERENCES <gerefereerde tabel> (<kolomnamen>);
```

## 11.3 CREATE TABLE ... FOREIGN KEY

We kunnen een Foreign Key constraint ook tegelijkertijd aanmaken met het aanmaken van de tabel zelf met de SQL syntax:

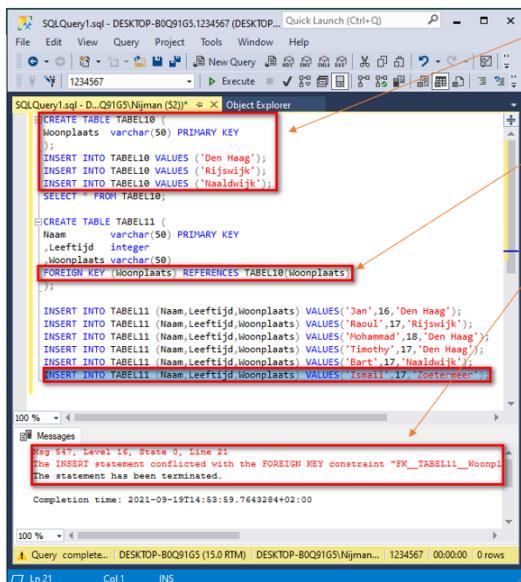
```
CREATE TABLE <tabelnaam> (<kolomnaam1 datatype>, <kolomnaam2 datatype>,...)
FOREIGN KEY (<kolomnamen>) REFERENCES <tabelnaam gerefereerde tabel> (<kolomnamen>);
```

Als voorbeeld maken we een tabel Tabel10 met daarin de mogelijke woonplaatsen en een tweede tabel Tabel11 met personen die refereert naar de tabel Tabel10 met woonplaatsen:

```
CREATE TABLE TABEL10 (Woonplaats varchar(50) PRIMARY KEY);
INSERT INTO TABEL10 VALUES ('Den Haag');
INSERT INTO TABEL10 VALUES ('Rijswijk');
INSERT INTO TABEL10 VALUES ('Naaldwijk');
SELECT * FROM TABEL10;
```

```
CREATE TABLE TABEL11 (Naam varchar(50) PRIMARY KEY, Leeftijd integer, Woonplaats varchar(50)
FOREIGN KEY (Woonplaats) REFERENCES TABEL10(Woonplaats));
INSERT INTO TABEL11 (Naam, Leeftijd, Woonplaats) VALUES('Jan', 16, 'Den Haag');
INSERT INTO TABEL11 (Naam, Leeftijd, Woonplaats) VALUES('Raoul', 17, 'Rijswijk');
INSERT INTO TABEL11 (Naam, Leeftijd, Woonplaats) VALUES('Mohammad', 18, 'Den Haag');
INSERT INTO TABEL11 (Naam, Leeftijd, Woonplaats) VALUES('Timothy', 17, 'Den Haag');
INSERT INTO TABEL11 (Naam, Leeftijd, Woonplaats) VALUES('Bart', 17, 'Naaldwijk');
INSERT INTO TABEL11 (Naam, Leeftijd, Woonplaats) VALUES('Ismail', 17, 'Zoetermeer');
SELECT * FROM TABEL11;
```

Het resultaat van deze Foreign Key constraint is hieronder afgebeeld:



```

CREATE TABLE TABEL10 (
    Woonplaats varchar(50) PRIMARY KEY
);
INSERT INTO TABEL10 VALUES ('Den Haag');
INSERT INTO TABEL10 VALUES ('Rijswijk');
INSERT INTO TABEL10 VALUES ('Naaldwijk');
SELECT * FROM TABEL10;

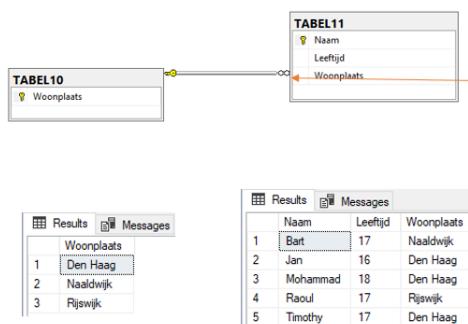
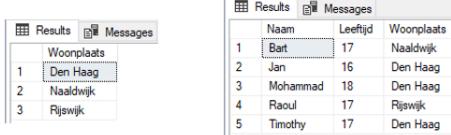
CREATE TABLE TABEL11 (
    Naam      varchar(50) PRIMARY KEY
    ,Leeftijd integer
    ,Woonplaats varchar(50)
    FOREIGN KEY (Woonplaats) REFERENCES TABEL10(Woonplaats)
);
INSERT INTO TABEL11 (Naam, Leeftijd, Woonplaats) VALUES ('Jan', 16, 'Den Haag');
INSERT INTO TABEL11 (Naam, Leeftijd, Woonplaats) VALUES ('Raoul', 17, 'Rijswijk');
INSERT INTO TABEL11 (Naam, Leeftijd, Woonplaats) VALUES ('Mohammad', 18, 'Den Haag');
INSERT INTO TABEL11 (Naam, Leeftijd, Woonplaats) VALUES ('Timothy', 17, 'Den Haag');
INSERT INTO TABEL11 (Naam, Leeftijd, Woonplaats) VALUES ('Bart', 17, 'Naaldwijk');
INSERT INTO TABEL11 (Naam, Leeftijd, Woonplaats) VALUES ('Ismael', 17, 'Zoetermeer');

Msg 547, Level 16, State 0, Line 21
The INSERT statement conflicted with the
FOREIGN KEY constraint
"FK_TABEL11_Woonplaats". The
conflict occurred in database "1234567",
table "dbo.TABEL10", column 'Woonplaats'.
The statement has been terminated.

Completion time: 2021-09-19T14:59:59.7643284+02:00

```

- Creëer tabel TABEL10 met de mogelijke Woonplaatsen en de **Primary Key Constraint** op de kolom Woonplaats
- Creëer tabel TABEL11 met een **Foreign Key Constraint** naar de **Primary Key Constraint** van tabel TABEL10
- Bij het invoeren van een persoon met woonplaats Zoetermeer geeft de **Foreign Key Constraint** een foutmelding:  
 Msg 547, Level 16, State 0, Line 21 The  
 INSERT statement conflicted with the  
 FOREIGN KEY constraint  
 "FK\_TABEL11\_Woonplaats". The  
 conflict occurred in database "1234567",  
 table "dbo.TABEL10", column 'Woonplaats'.  
 The statement has been terminated.
- Zoek via Google naar oplossingen voor melding **Msg 547**
- De **Foreign Key Constraint** voorkomt dat in TABEL11 een woonplaats ingevuld kan worden die niet staat in TABEL10
- Er bestaat een **Foreign Key Relation** tussen de kolom Woonplaats van TABEL11 en de kolom Woonplaats van de TABEL10
- Een **Foreign Key Relation** wordt ook wel een **Parent Child Relation** genoemd daar de **parents** moeten bestaan voordat een **child** kan bestaan
- Een invoer van een woonplaats die niet in TABEL10 voorkomt zal door de **Foreign Key Constraint** geweigerd worden en een foutmelding geven
- Een **Foreign Key Constraint** kan ook worden gemaakt naar een **Candidate Key (Unique NOT NULL)** i.p.v. naar de **Primary key**

Results	Messages																		
<table border="1"> <thead> <tr> <th>Woonplaats</th> </tr> </thead> <tbody> <tr><td>Den Haag</td></tr> <tr><td>Naaldwijk</td></tr> <tr><td>Rijswijk</td></tr> </tbody> </table>	Woonplaats	Den Haag	Naaldwijk	Rijswijk															
Woonplaats																			
Den Haag																			
Naaldwijk																			
Rijswijk																			
	<table border="1"> <thead> <tr> <th>Naam</th> <th>Leeftijd</th> <th>Woonplaats</th> </tr> </thead> <tbody> <tr><td>Bart</td><td>17</td><td>Naaldwijk</td></tr> <tr><td>Jan</td><td>16</td><td>Den Haag</td></tr> <tr><td>Mohammad</td><td>18</td><td>Den Haag</td></tr> <tr><td>Raoul</td><td>17</td><td>Rijswijk</td></tr> <tr><td>Timothy</td><td>17</td><td>Den Haag</td></tr> </tbody> </table>	Naam	Leeftijd	Woonplaats	Bart	17	Naaldwijk	Jan	16	Den Haag	Mohammad	18	Den Haag	Raoul	17	Rijswijk	Timothy	17	Den Haag
Naam	Leeftijd	Woonplaats																	
Bart	17	Naaldwijk																	
Jan	16	Den Haag																	
Mohammad	18	Den Haag																	
Raoul	17	Rijswijk																	
Timothy	17	Den Haag																	

Een Foreign Key constraint kan op verschillende manieren worden toegepast. Zo kan een FOREIGN KEY constraint bijvoorbeeld naar 1 kolom van een andere tabel verwijzen:

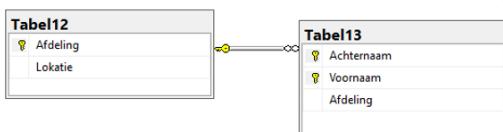
## Foreign Key Constraints 1

```

CREATE TABLE Tabel12 (
    Afdeling VARCHAR(10),
    Lokatie VARCHAR(10)
    CONSTRAINT PK_Tabel12 PRIMARY KEY (Afdeling)
);

CREATE TABLE Tabel13 (
    Achternaam VARCHAR(20),
    Voornaam VARCHAR(20),
    Afdeling VARCHAR(10)
    CONSTRAINT PK_Tabel13 PRIMARY KEY (Achternaam, Voornaam),
    CONSTRAINT FK_Tabel13_Afdeling FOREIGN KEY (Afdeling) REFERENCES Tabel12 (Afdeling)
);

```



- Een **Foreign Key** is een kolom of combinatie van kolommen die refereert naar de **Primaire Key** in een andere tabel of de eigen tabel.
- In het voorbeeld hiernaast wordt vanuit **Tabel13 kolom Afdeling** gerefereerd naar **Tabel12 kolom Afdeling**.
- Het **Entity Relation Diagram** ziet er als volgt uit

Een FOREIGN KEY constraint kan ook naar een combinatie van kolommen in een andere tabel verwijzen:

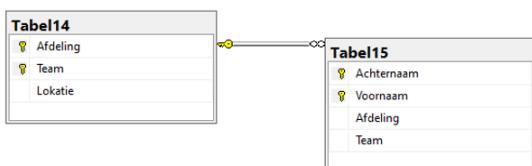
## Foreign Key Constraints 2

```

CREATE TABLE Tabel14 (
    Afdeling VARCHAR(10),
    Team VARCHAR(10),
    Lokatie VARCHAR(10)
    CONSTRAINT PK_Tabel14 PRIMARY KEY (Afdeling,Team)
);

CREATE TABLE Tabel15 (
    Achternaam VARCHAR(20),
    Voornaam VARCHAR(20),
    Afdeling VARCHAR(10),
    Team VARCHAR(10)
    CONSTRAINT PK_Tabel15 PRIMARY KEY (Achternaam, Voornaam),
    CONSTRAINT FK_Tabel15_Afdeling FOREIGN KEY (Afdeling,Team) REFERENCES Tabel14 (Afdeling,Team)
);

```

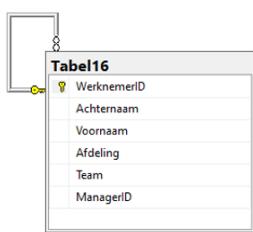


- Een **Foreign Key** is een kolom of combinatie van kolommen die refereert naar de **Primaire Key** in een andere tabel of de eigen tabel.
- In het voorbeeld hiernaast wordt vanuit **Tabel15 kolomcombinatie (Afdeling,Team)** gerefereerd naar **Tabel14 kolomcombinatie (Afdeling,Team)**.
- Dit wordt ook wel **Composite Primary Key** (Samengestelde Primaire Sleutel) genoemd.
- Het **Entity Relation Diagram** ziet er als volgt uit

Een FOREIGN KEY constraint kan ook naar de PRIMARY KEY van de tabel zelf verwijzen:

## Foreign Key Constraints 3

```
CREATE TABLE Tabel16 (
WerknemerID int,
Achternaam VARCHAR(20),
Voornaam VARCHAR(20),
Afdeling VARCHAR(10),
Team VARCHAR(10),
ManagerID INT
CONSTRAINT PK_Tabel16 PRIMARY KEY (WerknemerID),
CONSTRAINT FK_Tabel16_Manager FOREIGN KEY (ManagerID) REFERENCES Tabel16 (WerknemerID);
```

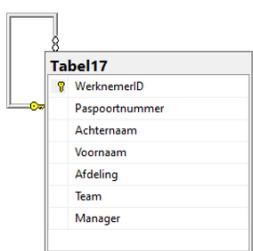


- Een **Foreign Key** is een kolom of combinatie van kolommen die refereert naar de **Primaire Key** in een andere tabel of de eigen tabel.
- In het voorbeeld hiernaast wordt vanuit **Tabel16 kolom (ManagerID)** gerefereerd naar **Tabel16 kolom (WerknemerID)**.
- De Foreign Key refereert dus naar de **Primary Key van de tabel zelf!** De manager is zelf immers ook een werknemer van het bedrijf.
- Dit wordt ook wel **Self Referencing Table** genoemd.
- Het **Entity Relation Diagram** ziet er als volgt uit

Een FOREIGN KEY constraint kan ook naar een andere sleutel dan de PRIMARY KEY (ALTERNATE KEY) van een andere tabel of zichzelf verwijzen:

## Foreign Key Constraints 4

```
CREATE TABLE Tabel17 (
WerknemerID int,
Paspoortnummer VARCHAR(10) NOT NULL UNIQUE,
Achternaam VARCHAR(20),
Voornaam VARCHAR(20),
Afdeling VARCHAR(10),
Team VARCHAR(10),
Manager VARCHAR(10)
CONSTRAINT PK_Tabel17 PRIMARY KEY (WerknemerID),
CONSTRAINT FK_Tabel17_Manager FOREIGN KEY (Manager) REFERENCES Tabel17 (Paspoortnummer);
```



- Een **Foreign Key** is een kolom of combinatie van kolommen die refereert naar de **Primaire Key** in een andere tabel of de eigen tabel.
- Maar het is ook mogelijk om de **Foreign key** kolom(men) te laten refereren naar een **Alternate Key** (ook wel **Secondary Key**) i.p.v. naar de **Primary Key** van de tabel (zelf).
- Een **Alternate Key** is een Key die ook **UNIQUE** en **NOT NULL** is (ook wel **Kandidate Key**) maar **niet gekozen** is tot de **Primary Key** van de tabel.
- In het voorbeeld hiernaast wordt vanuit **Tabel17 kolom Manager** gerefereerd naar **Tabel17 kolom Paspoortnummer van dezelfde tabel** i.p.v. Primary Key kolom **WerknemerID**.
- Het **Entity Relation Diagram** ziet er als volgt uit

Achter de tabelnaam waarnaar gerefereerd wordt mag een kolomnaam (of een verzameling kolomnamen) worden gespecificeerd. Als deze weg gelaten wordt dan moet de primaire sleutel (Primary Key) van de gerefereerde tabel overeen komen met de refererende sleutel (Foreign Key).

Een NULL waarde in de refererende sleutel is toegestaan, ook al kan de primaire sleutel zelf nooit een NULL waarde bevatten. De Foreign Key constraint zal dus géén controle uitvoeren of er een NULL waarde ingevoerd wordt, daar de Primary Key constraint van de gerefereerde tabel reeds uitsluit dat

er een NULL waarde in de refererende kolom kan worden ingevoerd. Immers indien we in bovenstaand voorbeeld in Tabel11 een persoon willen invoeren zonder woonplaats, dan zal de Foreign Key constraint de actie reeds weigeren daar er in de kolom woonplaats van Tabel10 géén NULL waarde kan staan door zijn Primary Key constraint.

Het aantal en de datatypen van de kolommen in de refererende sleutel moet gelijk zijn aan het aantal en de datatypen van de kolommen in de primaire sleutel van de gerefereerde tabel. Bij voorbeeld een Foreign Key op de refererende tabel met 2 kolommen kan alleen refereren naar een combinatie van 2 kolommen op de gerefereerde tabel.

## 11.4 ON UPDATE/DELETE action

Elke specificatie van een Foreign Key constraint bestaat uit drie delen:

**CREATE TABLE ... / ALTER TABLE ... ADD CONSTRAINT ...**

**FOREIGN KEY (<kolomnamen>) REFERENCES <tabelnaam gerefereerde tabel> (<kolomnamen>) ON UPDATE <actie> ON DELETE <actie>;**

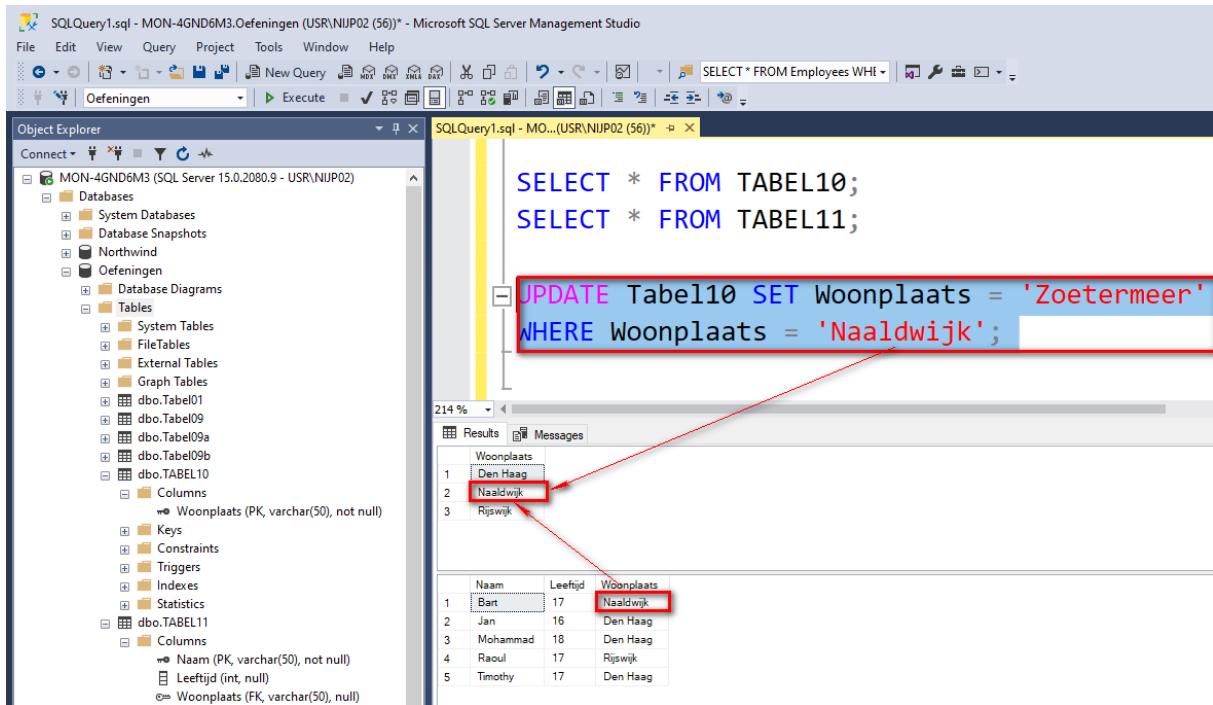
- In het eerste deel wordt aangegeven welke kolom (of combinatie van kolommen) de refererende sleutel vormt (**FOREIGN KEY (<kolomnamen>)**).
- Met het tweede deel geven we aan naar welke tabel en kolommen de refererende sleutel wijst (**REFERENCES <tabelnaam gerefereerde tabel> (<kolomnamen>)**).
- Het derde deel is de refererende actie (is soms afwezig) **ON UPDATE <actie> ON DELETE <actie>;**

Bij het invoeren van een waarde in de refererende tabel controleert de FOREIGN KEY constraint of de waarde reeds in de gerefereerde tabel zit. Als gevolg hiervan weten we zeker dat alle waarden in de refererende tabel voorkomen in de gerefereerde tabel.

Maar wat gebeurt er nu als we een waarde in de gerefereerde tabel wijzigen of verwijderen? Er wordt dan een waarde gewijzigd terwijl er nog verwijzingen staan naar de oude waarde in de gerefereerde tabel. In onderstaand voorbeeld willen we het volgende SQL commando uitvoeren:

```
UPDATE Tabel10 SET Woonplaats = 'Zoetermeer'  
WHERE Woonplaats = 'Naaldwijk';
```

We willen dus in Tabel10 de waarde Naaldwijk veranderen in de waarde Zoetermeer terwijl er vanuit Tabel11 een verwijzing bestaat naar de waarde Naaldwijk. Krijgen we dan een situatie waarbij er in Tabel11 een waarde staat die niet meer in Tabel10 staat?



```

SELECT * FROM TABEL10;
SELECT * FROM TABEL11;

UPDATE Tabel10 SET Woonplaats = 'Zoetermeer'
WHERE Woonplaats = 'Naaldwijk';

```

	Woonplaats
1	Den Haag
2	Naaldwijk
3	Rijswijk

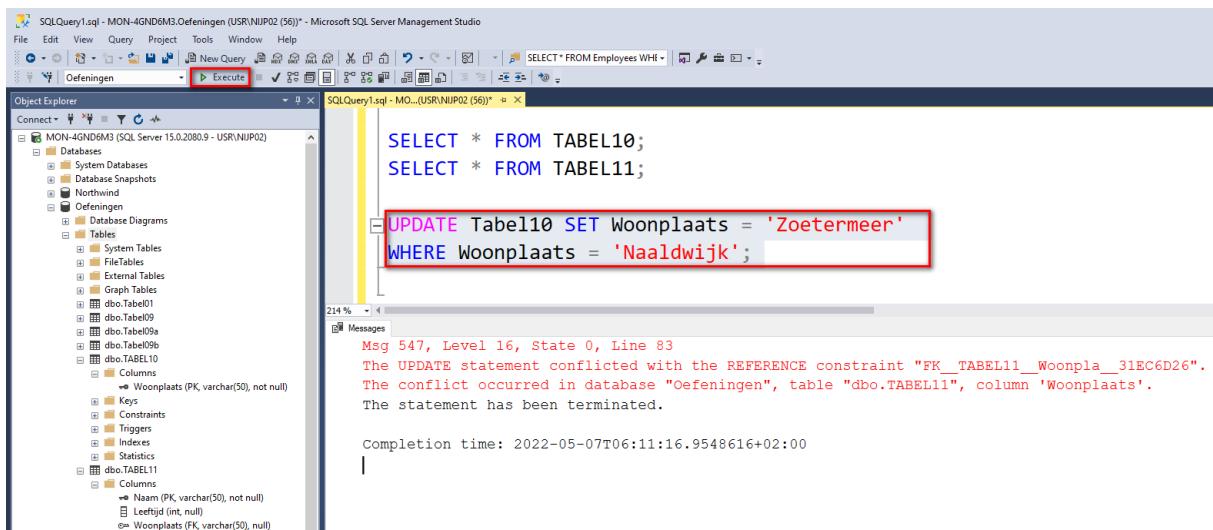
	Naam	Leeftijd	Woonplaats
1	Bart	17	Naaldwijk
2	Jan	16	Den Haag
3	Mohammad	18	Den Haag
4	Raoul	17	Rijswijk
5	Timothy	17	Den Haag

Wanneer we het uitvoeren dan blijkt dat SQL Server default (standaard) het commando niet toelaat en de actie weigert met een foutmelding, zoals afgebeeld in onderstaand voorbeeld:

```

Msg 547, Level 16, State 0, Line 83
The UPDATE statement conflicted with the REFERENCE constraint
"FK_TABEL11_Woonpla_31EC6D26".
The conflict occurred in database "Oefeningen", table "dbo.TABEL11", column
'Woonplaats'.
The statement has been terminated.

```



```

SELECT * FROM TABEL10;
SELECT * FROM TABEL11;

UPDATE Tabel10 SET Woonplaats = 'Zoetermeer'
WHERE Woonplaats = 'Naaldwijk';

```

Messages		
Msg 547, Level 16, State 0, Line 83 The UPDATE statement conflicted with the REFERENCE constraint "FK_TABEL11_Woonpla_31EC6D26". The conflict occurred in database "Oefeningen", table "dbo.TABEL11", column 'Woonplaats'. The statement has been terminated.		

M.b.v. een refererende actie **ON UPDATE** en **ON DELETE** kunnen we echter het default (standaard) gedrag van de FOREIGN KEY constraint aanpassen, door bij het creëren van de constraint aan te geven hoe de constraint zich moet gedragen bij een UPDATE (**ON UPDATE**) of hoe de constraint zich moet gedragen bij een DELETE (**ON DELETE**) actie:

- **NO ACTION**

De waarde in de gerefereerde tabel (Tabel10) wordt niet veranderd zolang er nog verwijzingen naar zijn en er wordt een foutmelding gegeven. Dit is dus het default (standaard) gedrag van SQL Server.

In bovenstaand voorbeeld zou dat betekenen dat in Tabel10 en Tabel11 **géén wijzigingen** worden gemaakt en er een foutmelding komt.

- **CASCADE**

De waarde in de gerefereerde tabel (Tabel10) wordt veranderd en de waarden in de refererende tabel (Tabel11) veranderen mee met de veranderingen in de gerefereerde tabel (Tabel10).

In bovenstaand voorbeeld zou dat betekenen dat in Tabel10 de waarde Naaldwijk wordt veranderd in de waarde Zoetermeer maar dat ook in Tabel11 alle waarden Naaldwijk worden veranderd in Zoetermeer.

- **SET NULL**

De waarde in de gerefereerde tabel (Tabel10) wordt veranderd, maar de verwijzingen vanuit de refererende tabel (Tabel11) worden leeg (NULL) gemaakt.

In bovenstaand voorbeeld zou dat betekenen dat in Tabel10 de waarde Naaldwijk wordt veranderd in Zoetermeer maar dat in Tabel11 alle waarden Naaldwijk leeg (NULL) worden gemaakt.

- **SET DEFAULT**

De waarde in de gerefereerde tabel (Tabel10) wordt veranderd, maar de verwijzingen vanuit de refererende tabel (Tabel11) worden veranderd naar een default waarde (die bij het maken van Tabel11 is opgegeven met het woord **DEFAULT** achter de kolom).

In bovenstaand voorbeeld zou dat betekenen dat in Tabel10 de waarde Naaldwijk wordt veranderd in Zoetermeer maar dat ook in Tabel11 alle waarden Naaldwijk worden veranderd in een default waarde (die bij het creëren van Tabel11 is opgegeven).

Met deze refererende acties kunnen we dus voorkomen dat de gegevens in de tabellen niet meer voldoen aan de Foreign Key constraint.

## Foreign Key Actions

```

CREATE TABLE Tabel119 (
    Achternaam VARCHAR(20),
    Voornaam VARCHAR(20),
    Afdeling VARCHAR(10)
    CONSTRAINT PK_Tabel119 PRIMARY KEY (Achternaam, Voornaam),
    CONSTRAINT FK_Tabel119_Afdeling FOREIGN KEY (Afdeling) REFERENCES Tabel18 (Afdeling)
    ON UPDATE CASCADE
    ON DELETE CASCADE
);

CREATE TABLE Tabel119 (
    Achternaam VARCHAR(20),
    Voornaam VARCHAR(20),
    Afdeling VARCHAR(10)
    CONSTRAINT PK_Tabel119 PRIMARY KEY (Achternaam, Voornaam),
    CONSTRAINT FK_Tabel119_Afdeling FOREIGN KEY (Afdeling) REFERENCES Tabel18 (Afdeling)
    ON UPDATE SET NULL
    ON DELETE SET NULL
);

CREATE TABLE Tabel119 (
    Achternaam VARCHAR(20),
    Voornaam VARCHAR(20),
    Afdeling VARCHAR(10)
    CONSTRAINT PK_Tabel119 PRIMARY KEY (Achternaam, Voornaam),
    CONSTRAINT FK_Tabel119_Afdeling FOREIGN KEY (Afdeling) REFERENCES Tabel18 (Afdeling)
    ON UPDATE SET DEFAULT
    ON DELETE SET DEFAULT
);

CREATE TABLE Tabel119 (
    Achternaam VARCHAR(20),
    Voornaam VARCHAR(20),
    Afdeling VARCHAR(10)
    CONSTRAINT PK_Tabel119 PRIMARY KEY (Achternaam, Voornaam),
    CONSTRAINT FK_Tabel119_Afdeling FOREIGN KEY (Afdeling) REFERENCES Tabel18 (Afdeling)
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
);

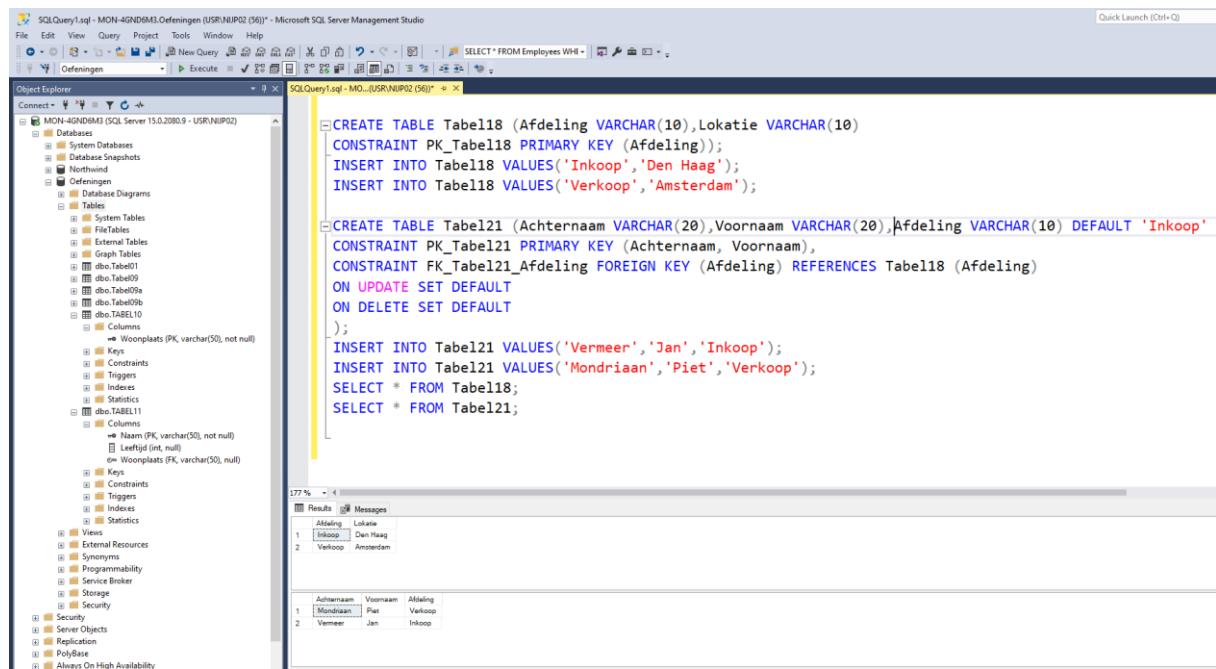
```

Met behulp van een **Foreign Key Action** (Refererende Actie) kunnen we het gedrag van een **Foreign Key Constraint** veranderen.

Een **Foreign Key Action** bestaat uit 2 delen:

1. Voor welke instructie de refererende actie geldt: **UPDATE, DELETE**
2. Specificatie van de te nemen actie als waarde in **Parent Table** wordt aangepast:
  - CASCADE** = Waarden in refererende tabel worden ook gewijzigd
  - SET NULL** = Waarden in refererende tabel worden op NULL gezet (mits Nullable)
  - SET DEFAULT** = Waarden in refererende tabel worden op default waarde gezet
  - NO ACTION (default)** = Er ontstaat er een foutmelding en actie wordt terug gedraaid

In onderstaand voorbeeld heeft de kolom Afdeling in Tabel21 een DEFAULT waarde gekregen van 'Inkoop' en in de Foreign Key Constraint wordt de refererende actie ON UPDATE/DELETE SET DEFAULT gegeven, wat betekent dat de waarde van de kolom Afdeling 'Inkoop' wordt zodra er in Tabel 18 een waarde in kolom Afdeling wordt veranderd of verwijderd.



```

CREATE TABLE Tabel18 (Afdeling VARCHAR(10),Lokatie VARCHAR(10))
CONSTRAINT PK_Tabel18 PRIMARY KEY (Afdeling);
INSERT INTO Tabel18 VALUES('Inkoop','Den Haag');
INSERT INTO Tabel18 VALUES('Verkoop','Amsterdam');

CREATE TABLE Tabel21 (Achternaam VARCHAR(20),Voornaam VARCHAR(20),Afdeling VARCHAR(10) DEFAULT 'Inkoop'
CONSTRAINT PK_Tabel21 PRIMARY KEY (Achternaam, Voornaam),
CONSTRAINT FK_Tabel21_Afdeling FOREIGN KEY (Afdeling) REFERENCES Tabel18 (Afdeling)
ON UPDATE SET DEFAULT
ON DELETE SET DEFAULT
);

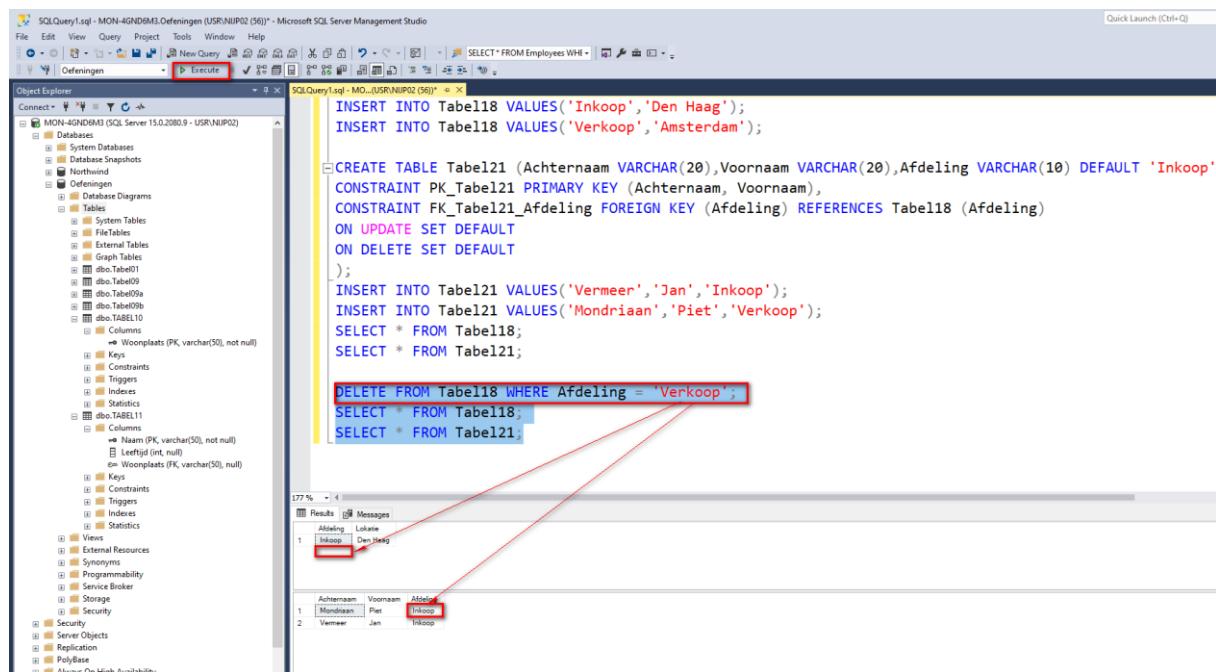
INSERT INTO Tabel21 VALUES('Vermeer','Jan','Inkoop');
INSERT INTO Tabel21 VALUES('Mondriaan','Piet','Verkoop');
SELECT * FROM Tabel18;
SELECT * FROM Tabel21;

```

Afdeling	Lokatie
Inkoop	Den Haag
Verkoop	Amsterdam

Achternaam	Voornaam	Afdeling
Mondriaan	Piet	Verkoop
Vermeer	Jan	Inkoop

We verwijderen nu de rij waarin in kolom Afdeling de waarde 'Verkoop' staat. Als gevolg daarvan wordt de betreffende rij in Tabel 18 daadwerkelijk verwijderd en worden de verwijzing in Tabel 21 veranderd van 'Verkoop' naar de default 'Inkoop'. Piet Mondriaan was dus in Tabel21 in afdeling Verkoop maar zit na de delete actie op Tabel18 in Tabel21 in de afdeling Inkoop:



```

INSERT INTO Tabel18 VALUES('Inkoop','Den Haag');
INSERT INTO Tabel18 VALUES('Verkoop','Amsterdam');

CREATE TABLE Tabel21 (Achternaam VARCHAR(20),Voornaam VARCHAR(20),Afdeling VARCHAR(10) DEFAULT 'Inkoop'
CONSTRAINT PK_Tabel21 PRIMARY KEY (Achternaam, Voornaam),
CONSTRAINT FK_Tabel21_Afdeling FOREIGN KEY (Afdeling) REFERENCES Tabel18 (Afdeling)
ON UPDATE SET DEFAULT
ON DELETE SET DEFAULT
);

INSERT INTO Tabel21 VALUES('Vermeer','Jan','Inkoop');
INSERT INTO Tabel21 VALUES('Mondriaan','Piet','Verkoop');
SELECT * FROM Tabel18;
SELECT * FROM Tabel21;

DELETE FROM Tabel18 WHERE Afdeling = 'Verkoop';
SELECT * FROM Tabel18;
SELECT * FROM Tabel21;

```

Afdeling	Lokatie
Inkoop	Den Haag

Achternaam	Voornaam	Afdeling
Mondriaan	Piet	Inkoop
Vermeer	Jan	Inkoop

We kunnen dus met de refererende acties de gegevens in de refererende tabel naar wens mee laten veranderen bij een verandering in de gerefereerde tabel.

## 11.5 Voorbeeld van toepassing van een Foreign Key Constraint

In de database CLUB wordt de tabel WOONPLAATS aangemaakt en gevuld met de volgende SQL-statements:

```
CREATE TABLE WOONPLAATS (Plaats varchar(50) PRIMARY KEY, Provincie varchar(50));
INSERT INTO WOONPLAATS VALUES ('Den Haag', 'Zuid Holland');
INSERT INTO WOONPLAATS VALUES ('Rijswijk', 'Zuid Holland');
INSERT INTO WOONPLAATS VALUES ('Naaldwijk', 'Zuid Holland');
```

Daarna wordt in de database CLUB de tabel PERSOON aangemaakt met het volgende statement:

```
CREATE TABLE PERSOON (Naam varchar(50) PRIMARY KEY, Leeftijd integer, Plaats varchar(50)
FOREIGN KEY (Plaats) REFERENCES WOONPLAATS(Plaats));
```

Vervolgens worden achtereenvolgens de onderstaande SQL-statements uitgevoerd.

1. **INSERT INTO PERSOON (Naam, Leeftijd, Plaats) VALUES('Jan', 16, 'Den Haag');**
2. **INSERT INTO PERSOON (Naam, Leeftijd, Plaats) VALUES('Raoul', 17, 'Rijswijk');**
3. **INSERT INTO PERSOON (Naam, Leeftijd, Plaats) VALUES('Mohammad', 18, 'Den Haag');**
4. **INSERT INTO PERSOON (Naam, Leeftijd, Plaats) VALUES('Timothy', 17, 'Den Haag');**
5. **INSERT INTO PERSOON (Naam, Leeftijd, Plaats) VALUES('Bart', 17, 'Naaldwijk');**
6. **INSERT INTO PERSOON (Naam, Leeftijd, Plaats) VALUES('Ismail', 17, 'Zoetermeer');**
7. **INSERT INTO WOONPLAATS (Plaats) VALUES ('Delft');**
8. **INSERT INTO WOONPLAATS VALUES ('Delft', 'Zuid Holland');**
9. **INSERT INTO PERSOON (Naam, Leeftijd, Plaats) VALUES('Jeroen', 17, 'Delft');**
10. **INSERT INTO WOONPLAATS VALUES ('Zoetermeer', 'Limburg');**
11. **INSERT INTO PERSOON (Naam, Leeftijd, Plaats) VALUES('Timothy', 17, 'Zoetermeer');**

De volgende INSERTS zullen een foutmelding geven:

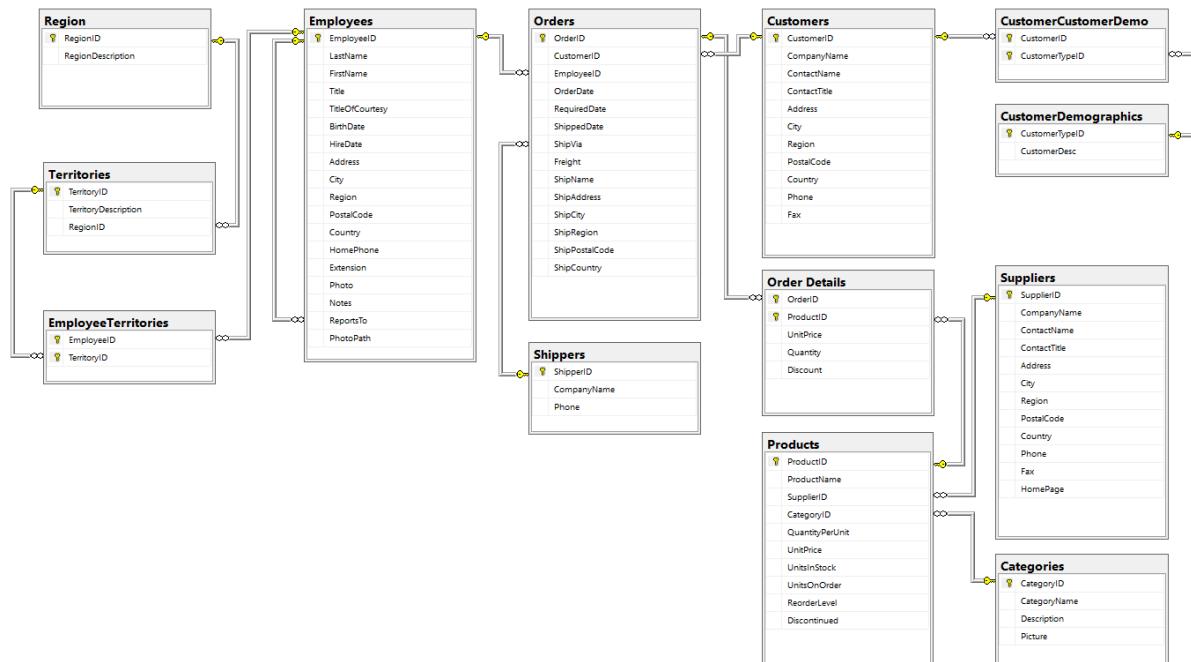
INSERT 6 zal een foutmelding geven omdat Zoetermeer niet in tabel Woonplaats staat (Foreign Key constraint op tabel Persoon)

INSERT 8 zal een foutmelding geven omdat Delft reeds in INSERT 7 is ingevoerd (Primary Key constraint op tabel Woonplaats)

INSERT 11 zal een foutmelding geven omdat Timothy reeds in INSERT 4 is ingevoerd (Primary Key constraint op tabel Persoon)

## 11.6 FOREIGN KEYS in het Northwind datamodel

In hoofdstuk 6 hebben we het Northwind datamodel bestudeerd en hebben we de lijnen tussen de tabellen slechts relaties genoemd.



De kolommen in de tabellen met een sleutel symbool betreft de Primary Key constraint van de tabel.

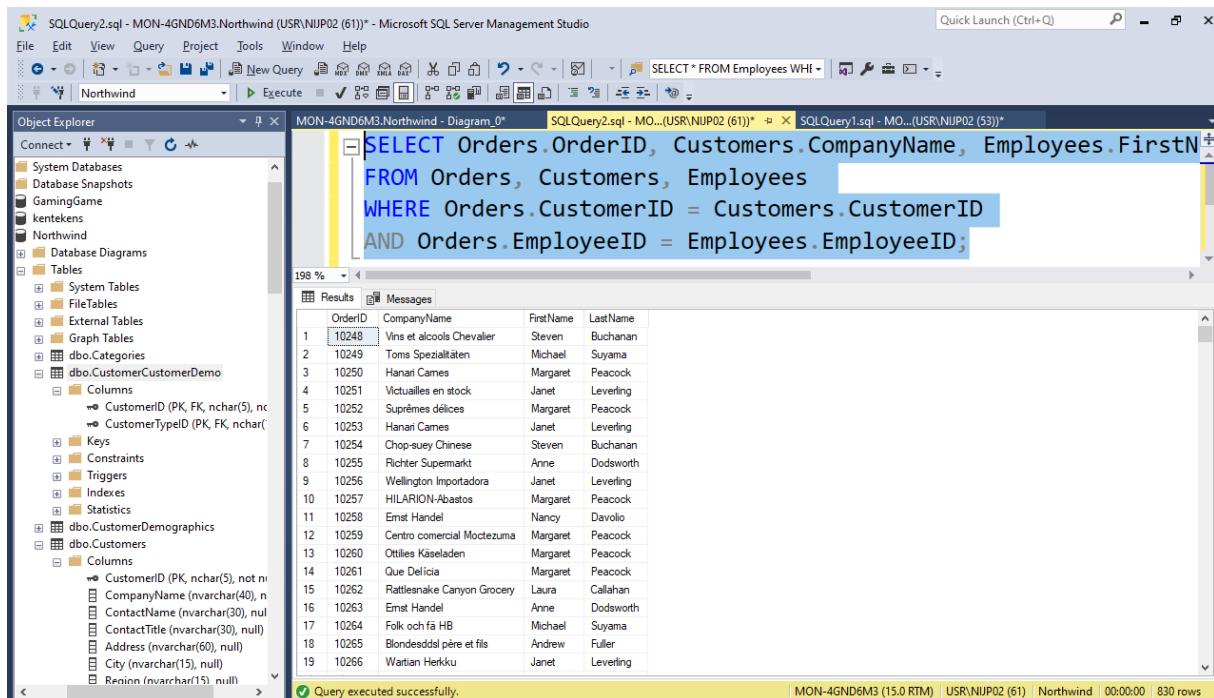
Met een lijn tussen de tabellen worden de Foreign Key constraints (Relaties) tussen de tabellen weergegeven. Aan de ene zijde van de Foreign Key constraint staat het oneindig teken die aangeeft dat er aan die zijde van de refererende tabel nul, één of meerdere rijen een relatie hebben en aan de andere zijde van de Foreign Key constraint staat een sleutel teken die aangeeft dat er aan die zijde gerefereerd wordt naar een sleutel (Primary or Alternate Key constraint).

In de praktijk wordt vrijwel altijd gerefereerd naar de Primary Key constraint van de gerefereerde tabel en komt het minder vaak voor dat er gerefereerd wordt naar een Alternate Key constraint van de gerefereerde tabel.

De tabel Orders in de database Northwind bevat de gegevens van de orders met daarin ook het CustomerID en het EmployeeID. Stel dat wij het OrderID samen met de CompanyName van de klant en de Voornaam en Achternaam van de verkoper in een rapport willen zien, dan zullen we de gegevens uit de tabel Orders moeten koppelen met de gegevens uit de tabellen Customers en Employees om dat deze gegevens niet in de tabel Orders zelf aanwezig zijn:

```

SELECT Orders.OrderID, Customers.CompanyName, Employees.FirstName, Employees.LastName
FROM Orders, Customers, Employees
WHERE Orders.CustomerID = Customers.CustomerID
AND Orders.EmployeeID = Employees.EmployeeID;
    
```



```

SELECT Orders.OrderID, Customers.CompanyName, Employees.FirstName
FROM Orders, Customers, Employees
WHERE Orders.CustomerID = Customers.CustomerID
AND Orders.EmployeeID = Employees.EmployeeID;

```

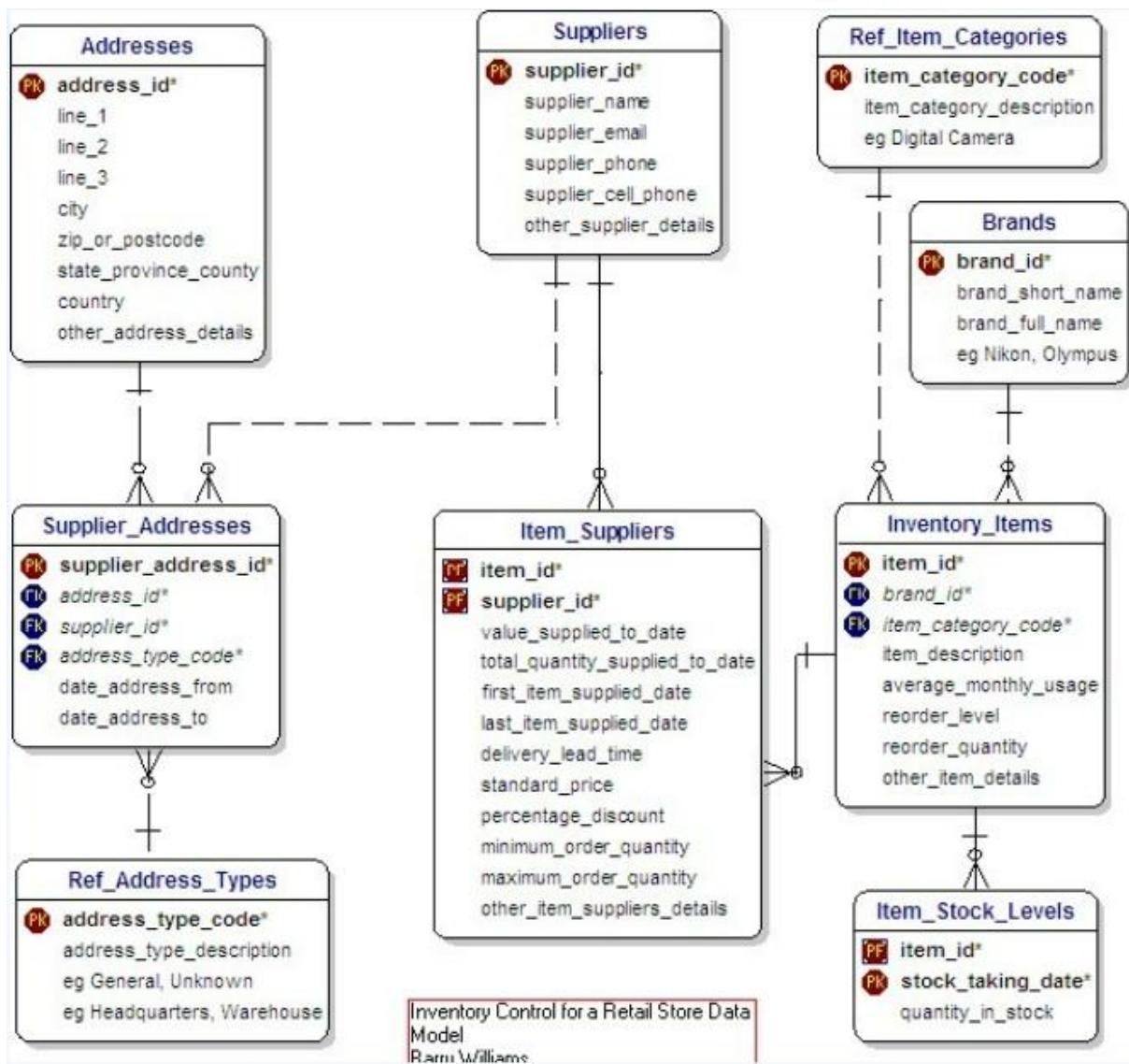
OrderID	CompanyName	FirstName	LastName
1	Vins et alcools Chevalier	Steven	Buchanan
2	Toms Spezialitäten	Michael	Suyama
3	Hanari Cames	Margaret	Peacock
4	Virtuailles en stock	Janet	Leverling
5	Suprèmes délices	Margaret	Peacock
6	Hanari Cames	Janet	Leverling
7	Chop-suey Chinese	Steven	Buchanan
8	Richter Supermarkt	Anne	Dodsworth
9	Wellington Importadora	Janet	Leverling
10	HILARION-Abastos	Margaret	Peacock
11	Ernst Handel	Nancy	Davolio
12	Centro comercial Móvilzuma	Margaret	Peacock
13	Ottilies Käseladen	Margaret	Peacock
14	Que Delicia	Margaret	Peacock
15	Rattlesnake Canyon Grocery	Laura	Callahan
16	Ernst Handel	Anne	Dodsworth
17	Folk och få HB	Michael	Suyama
18	Blondesdåhl père et fils	Andrew	Fuller
19	Wartian Herkku	Janet	Leverling

Bovenstaande query waarin de tabellen via de Foreign Key constraints met elkaar worden gekoppeld heet een (impliciete) JOIN, welke in hoofdstuk 15 inhoudelijk wordt behandeld.

Door de Foreign Key constraints kunnen gegevens in tabel Orders pas ingevoerd worden nadat de gegevens waarnaar gerefereerd word in de tabellen Customers en Employees zijn ingevoerd. Het is dus niet mogelijk om een order in te voeren in tabel Orders zolang de klant en de medewerker nog niet zijn ingevoerd in de tabellen Customers en Employees.

## 11.7 FOREIGN KEYS in Entity Relation Diagram

Stel dat je een lege database met onderstaand Entity Relation Diagram moet vullen met gegevens. Op welke volgorde kan dat dan worden gedaan?



De lijnen tussen de tabellen zijn de Foreign Key constraints tussen de tabellen.

Als we naar het diagram kijken dan zien we het volgende:

- Tabel Supplier\_Addresses heeft Foreign Key constraints naar de tabellen Addresses, Suppliers en Ref\_Address\_Types. Dat wil zeggen dat de tabellen Addresses, Suppliers en Ref\_Address\_Types gevuld moeten zijn voordat de tabel Supplier\_Addresses wordt gevuld.
- Tabel Item\_Suppliers heeft Foreign Key constraints naar de tabellen Suppliers en Inventory\_Items. Dat wil zeggen dat de tabellen Suppliers en Inventory\_Items gevuld moeten zijn voordat de tabel Item\_Suppliers wordt gevuld.
- Tabel Inventory\_Items heeft Foreign Key constraints naar de tabellen Ref\_Item\_Categories en Brands. Dat wil zeggen dat de tabellen Ref\_Item\_Categories en Brands gevuld moeten zijn voordat tabel Inventory\_Items wordt gevuld.
- Tabel Item\_Stock\_Levels heeft een Foreign Key constraint naar de tabel Inventory\_Items. Dat wil zeggen dat de tabel Inventory\_Items gevuld moet zijn voordat de tabel Item\_Stock\_Levels wordt gevuld.

Mogelijke volgorde om de tabellen te vullen zijn bijvoorbeeld:

- Suppliers -> Addresses -> Ref\_Item\_Categories -> Brands -> Ref\_address\_Types -> Supplier\_Addresses -> Inventory\_Items -> Item\_Stock\_Levels -> Item\_Suppliers
- Addresses -> Suppliers -> Ref\_Address\_Types -> Supplier\_Addresses -> Ref\_Item\_Categories -> Brands -> Inventory\_Items -> Item\_Suppliers -> Item\_Stock\_Levels

Niet mogelijke volgorde om de tabellen te vullen zijn bijvoorbeeld:

- Item\_Suppliers -> Inventory\_Items -> Supplier\_Addresses -> Addresses -> Suppliers -> Ref\_Item\_Categories -> Brands -> Ref\_Address\_Types -> Item\_Stock\_Levels  
Omdat Inventory\_Items gevuld moet zijn voordat Item\_Suppliers kan worden gevuld
- Suppliers -> Addresses -> Ref\_Item\_Categories -> Brands -> Ref\_address\_Types -> Item\_Stock\_Levels -> Supplier\_Addresses -> Inventory\_Items -> Item\_Suppliers  
Omdat Inventory\_Items gevuld moet zijn voordat Item\_Stock\_Levels kan worden gevuld
- Ref\_Address\_Types -> Item\_Stock\_Levels -> Supplier\_Addresses -> Item\_Suppliers -> Inventory\_Items -> Brands -> Ref\_Item\_Categories -> Suppliers -> Addresses  
Omdat Inventory\_Items gevuld moet zijn voordat Item\_Stock\_Levels kan worden gevuld

## 11.8 Query voor het opvragen van de aanwezige FOREIGN KEY Constraints

We kunnen een lijst van al de aanwezige Foreign Key constraints maken met de volgende query:

USE Northwind;

```

SELECT obj.name AS FK_NAME,
    sch.name AS [schema_name],
    tab1.name AS [table],
    col1.name AS [column],
    tab2.name AS [referenced_table],
    col2.name AS [referenced_column]
FROM sys.foreign_key_columns fkc
INNER JOIN sys.objects obj
    ON obj.object_id = fkc.constraint_object_id
INNER JOIN sys.tables tab1
    ON tab1.object_id = fkc.parent_object_id
INNER JOIN sys.schemas sch
    ON tab1.schema_id = sch.schema_id
INNER JOIN sys.columns col1
    ON col1.column_id = parent_column_id AND col1.object_id = tab1.object_id
INNER JOIN sys.tables tab2
    ON tab2.object_id = fkc.referenced_object_id
INNER JOIN sys.columns col2
    ON col2.column_id = referenced_column_id AND col2.object_id = tab2.object_id;

```

FK_NAME	schema_name	table	column	referenced_table	referenced_column
FK_CustomerCustomerDemo	dbo	CustomerCustomerDemo	CustomerTypeID	CustomerDemographics	CustomerTypeID
FK_Territories_Region	dbo	Territories	RegionID	Region	RegionID
FK_Employees_Employees	dbo	Employees	ReportsTo	Employees	EmployeeID
FK_EmployeeTerritories_Employees	dbo	EmployeeTerritories	EmployeeID	Employees	EmployeeID
FK_Orders_Employees	dbo	Orders	EmployeeID	Employees	EmployeeID
FK_Products_Categories	dbo	Products	CategoryID	Categories	CategoryID
FK_CustomerCustomerDemo_Customers	dbo	CustomerCustomerDemo	CustomerID	Customers	CustomerID
FK_Orders_Customers	dbo	Orders	CustomerID	Customers	CustomerID
FK_Orders_Shippers	dbo	Orders	ShipVia	Shippers	ShipperID
FK_Products_Suppliers	dbo	Products	SupplierID	Suppliers	SupplierID
FK_Order_Details_Orders	dbo	Order Details	OrderID	Orders	OrderID
FK_Order_Details_Products	dbo	Order Details	ProductID	Products	ProductID
FK_EmployeeTerritories_Territories	dbo	EmployeeTerritories	TerritoryID	Territories	TerritoryID

Je hoeft de query nu nog niet te begrijpen, maar je kan de query kopiëren en naar SSMS plakken om de Foreign Key constraints in een database op te vragen.

## Hoofdstuk 12 CHECK Constraints

### 12.1 CHECK Constraints

Met Check Constraints kunnen we onze eigen bedrijfsspecifieke constraints maken.

Als voorbeeld maken we een tabel Tieners, waarin de toegestane leeftijden zijn 13,14,15,16,17,18,19 jaar. We hebben voor de leeftijd een CHECK constraint gemaakt om te controleren of de leeftijd wel tussen 13 t/m 19 jaar is.

```
CREATE TABLE TIENERS (Naam VARCHAR(20) PRIMARY KEY, Leeftijd INT  
CONSTRAINT CHECK_LEEFTIJD CHECK (Leeftijd > 12 AND Leeftijd < 20));
```

```
INSERT INTO Tieners VALUES ('Jan',13);  
INSERT INTO Tieners VALUES ('Piet',19);  
INSERT INTO Tieners VALUES ('Piet',15);  
INSERT INTO Tieners VALUES ('Klaas',12);  
INSERT INTO Tieners VALUES ('Erik',20);
```

De eerste twee INSERT acties zullen goed gaan omdat de personen Jan en Piet er nog niet in zitten.

De derde INSERT actie zal fout gaan omdat de persoon Piet er reeds in zit (Primary Key op tabel Tieners) en geeft een foutmelding:

```
Msg 2627, Level 14, State 1, Line 6  
Violation of PRIMARY KEY constraint 'PK_TIENERS_7375E70EDCF64751'. Cannot insert  
duplicate key in object 'dbo.TIENERS'. The duplicate key value is (Piet).  
The statement has been terminated.
```

De vierde INSERT actie zal fout gaan omdat de leeftijd van de persoon niet tussen 13 t/m 19 jaar is (CHECK constraint op tabel Tieners) en zal een foutmelding geven:

```
Msg 547, Level 16, State 0, Line 7  
The INSERT statement conflicted with the CHECK constraint "CHECK_LEEFTIJD". The  
conflict occurred in database "Oefeningen", table "dbo.TIENERS", column 'Leeftijd'.  
The statement has been terminated.
```

De vijfde INSERT actie zal fout gaan omdat de leeftijd van de persoon niet tussen 13 t/m 19 jaar is (CHECK constraint op tabel Tieners) en zal een foutmelding geven:

```
Msg 547, Level 16, State 0, Line 8  
The INSERT statement conflicted with the CHECK constraint "CHECK_LEEFTIJD". The  
conflict occurred in database "Oefeningen", table "dbo.TIENERS", column 'Leeftijd'.  
The statement has been terminated.
```

We kunnen ook een CHECK constraint aan een bestaande tabel toevoegen met de SQL Syntax:

```
ALTER TABLE <tabelnaam> ADD CONSTRAINT <constraintnaam> CHECK (<voorwaarde>);
```

Bijvoorbeeld:

```
ALTER TABLE TIENERS ADD CONSTRAINT CHECK_NAAM CHECK (Naam != 'Peter');  
INSERT INTO Tieners VALUES ('Peter',19);
```

Geeft een foutmelding omdat de naam Peter niet toegestaan is (CHECK constraint op tabel Tieners) en geeft een foutmelding:

```
Msg 547, Level 16, State 0, Line 12
```

The INSERT statement conflicted with the CHECK constraint "CHECK\_NAAM".  
The conflict occurred in database "Oefeningen", table "dbo.TIENERS", column 'Naam'.  
The statement has been terminated.

Met CHECK-constraints kunnen we dus onze eigen bedrijfsregels toevoegen. Elke clause die achter een WHERE kan worden gezet kan hiervoor ook gebruikt worden voor het maken van een CHECK-constraint.

Let wel op: Wanneer een constraint wordt gemaakt nadat de tabel is aangemaakt dan zal na het aanmaken van de constraint de constraint controleren of elke nieuwe waarde wel aan de constraint voldoet, echter controleert de toegevoegde constraint niet of de data die reeds in de tabel zit voldoet aan de voorwaarde. Wanneer we dus een constraint toevoegen aan een bestaande tabel dan is het mogelijk dat er reeds data in de tabel zit die niet voldoet aan de voorwaarden van de constraint!

## Hoofdstuk 13 SUB-QUERIES

Binnen een query mag een andere query aangeroepen worden. De aangeroepen query wordt een **subquery** genoemd. Andere namen voor subquery zijn **subselect** en **innerselect**. Het resultaat van een subquery wordt doorgegeven aan de aanroepende query die dan de verwerking kan vervolgen.

### 13.1 SUB-queries

Als we naar de inhoud van tabel Products in database Northwind kijken dan zien we dat er een product bestaat met het maximale aantal units in voorraad in het magazijn van 125.

**USE NORTHWIND;**  
**SELECT \* FROM PRODUCTS;**

	ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel	Discontinued
1	1	Chai	1	1	10 boxes x 20 bags	18.00	39	0	10	0
2	2	Chang	1	1	24 - 12 oz bottles	19.00	17	40	25	0
3	3	Aniseed Syrup	1	2	12 - 550 ml bottles	10.00	13	70	25	0
4	4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22.00	53	0	0	0
5	5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35	0	0	0	1
6	6	Grandma's Boysenberry Spread	3	2	12 - 8 oz jars	25.00	120	0	25	0
7	7	Uncle Bob's Organic Dried Pears	3	7	12 - 1 lb pkgs.	30.00	15	0	10	0
8	8	Northwoods Cranberry Sauce	3	2	12 - 12 oz jars	40.00	6	0	0	0
9	9	Mishi Kobe Niku	4	6	18 - 500 g pkgs.	97.00	29	0	0	1
10	10	Ikura	4	8	12 - 200 ml jars	31.00	31	0	0	0
11	11	Queso Cabrales	5	4	1 kg pkgs.	21.00	22	30	30	0
12	12	Queso Manchego La Pastora	5	4	10 - 500 g pkgs.	38.00	86	0	0	0
13	13	Konbu	6	8	2 kg box	6.00	24	0	5	0
14	14	Tofu	6	7	40 - 100 g pkgs.	23.25	35	0	0	0
15	15	Genen Shouyu	6	2	24 - 250 ml bottles	15.50	39	0	5	0
16	16	Pavlova	7	3	32 - 500 g boxes	17.45	29	0	10	0
17	17	Alice Mutton	7	6	20 - 1 kg tins	39.00	0	0	0	1
18	18	Carnarvon Tigers	7	8	16 kg pkgs.	62.50	42	0	0	0
19	19	Teatime Chocolate Biscuits	8	3	10 boxes x 12 pieces	9.20	25	0	5	0
20	20	Sir Rodney's Marmalade	8	3	30 gift boxes	81.00	40	0	0	0
21	21	Sir Rodney's Scones	8	3	24 pkgs. x 4 pieces	10.00	3	40	5	0
22	22	Gustaf's Knäckebrot	9	5	24 - 500 g pkgs.	21.00	104	0	25	0
23	23	Tunnbröd	9	5	12 - 250 g pkgs.	9.00	61	0	25	0
24	24	Guaraná Fantastica	10	1	12 - 355 ml cans	4.50	20	0	0	1
25	25	NuNuCa Nuß-Nougat-Creme	11	3	20 - 450 g glasses	14.00	76	0	30	0
26	26	Gumbar Gummibärchen	11	3	100 - 250 g bags	31.23	15	0	0	0
27	27	Schoggi Schokolade	11	3	100 - 100 g pieces	43.90	49	0	30	0
28	28	Rösle Sauerkraut	12	7	25 - 825 g cans	45.60	26	0	0	1
29	29	Thüringer Rostbratwurst	12	6	50 bags x 30 sausgs.	123.79	0	0	0	1
30	30	Nord-Ost Matjeshering	13	8	10 - 200 g glasses	25.89	10	0	15	0
31	31	Gorgonzola Telino	14	4	12 - 100 g pkgs.	12.50	0	70	20	0
32	32	Mascarpone Fabioli	14	4	24 - 200 g pkgs.	32.00	9	40	25	0
33	33	Geitost	15	4	500 g	2.50	112	0	20	0
34	34	Sasquatch Ale	16	1	24 - 12 oz bottles	14.00	111	0	15	0
35	35	Steeleye Stout	16	1	24 - 12 oz bottles	18.00	20	0	15	0
36	36	Inlagd Sill	17	8	24 - 250 g jars	19.00	112	0	20	0
37	37	Gravad lax	17	8	12 - 500 g pkgs.	26.00	11	50	25	0
38	38	Côte de Blaye	18	1	12 - 75 cl bottles	263.50	17	0	15	0
39	39	Chartreuse verte	18	1	750 cc per bottle	18.00	69	0	5	0
40	40	Boston Crab Meat	19	8	24 - 4 oz tins	18.40	123	0	30	0
41	41	Jack's New England Clam Chowder	19	8	12 - 12 oz cans	9.65	85	0	10	0
42	42	Singaporen Hokkien Fried Mee	20	5	32 - 1 kg pkgs.	14.00	26	0	0	1
43	43	Ipoh Coffee	20	1	16 - 500 g tins	46.00	17	10	25	0
44	44	Gula Malacca	20	2	20 - 2 kg bags	19.45	27	0	15	0
45	45	Rogede sild	21	8	1k pkgs.	9.50	5	70	15	0
46	46	Spegesild	21	8	4 - 450 g glasses	12.00	95	0	0	0
47	47	Zaanse koeken	22	3	10 - 4 oz boxes	9.50	36	0	0	0
48	48	Chocolate	22	3	10 pkgs.	12.75	15	70	25	0
49	49	Maxikaku	23	3	24 - 50 g pkgs.	20.00	10	60	15	0
50	50	Valkonen sulkaa	23	3	12 - 100 g bars	16.25	65	0	30	0
51	51	Manjimup Dried Apples	24	7	50 - 300 g pkgs.	53.00	20	0	10	0
52	52	Filo Mix	24	5	16 - 2 kg boxes	7.00	38	0	25	0
53	53	Perth Pasties	24	6	48 pieces	32.80	0	0	0	1
54	54	Tourière	25	6	16 pies	7.45	21	0	10	0
55	55	Pâté chinois	25	6	24 boxes x 2 pieces	24.00	115	0	20	0
56	56	Gnocchi di nonna Alice	26	5	24 - 250 g pkgs.	38.00	21	10	30	0
57	57	Ravioli Angelo	26	5	24 - 250 g pkgs.	19.50	36	0	20	0
58	58	Escargots de Bourgogne	27	8	24 pieces	13.25	62	0	20	0
59	59	Raclette Courdavault	28	4	5 kg pkgs.	55.00	79	0	0	0
60	60	Camembert Pierrot	28	4	15 - 300 g rounds	34.00	19	0	0	0
61	61	Sirop d'éralbe	29	2	24 - 500 ml bottles	28.50	113	0	25	0
62	62	Tarte au sucre	29	3	48 pies	49.30	17	0	0	0
63	63	Vegie-spread	7	2	15 - 625 g jars	43.90	24	0	5	0
64	64	Wimmers gute Semmelknödel	12	5	20 bags x 4 pieces	33.25	22	80	30	0
65	65	Louisiana Fiery Hot Pepper Sauce	2	2	32 - 8 oz bottles	21.05	76	0	0	0
66	66	Louisiana Hot Spiced Okra	2	2	24 - 8 oz jars	17.00	4	100	20	0
67	67	Laughing Lumberjack Lager	16	1	24 - 12 oz bottles	14.00	52	0	10	0
68	68	Scottish Longbreads	8	3	10 boxes x 8 pieces	12.50	6	10	15	0
69	69	Gudbrandsdalsost	15	4	10 kg pkgs.	36.00	26	0	15	0
70	70	Outback Lager	7	1	24 - 355 ml bottles	15.00	15	10	30	0
71	71	Flotemyost	15	4	10 - 500 g pkgs.	21.50	26	0	0	0
72	72	Mozzarella di Giovanni	14	4	24 - 200 g pkgs.	34.80	14	0	0	0
73	73	Röd Kaviar	17	8	24 - 150 g jars	15.00	101	0	5	0
74	74	Longlife Tofu	4	7	5 kg pkgs.	10.00	4	20	5	0
75	75	Rhönbräu Klosterbier	12	1	24 - 0.5 l bottles	7.75	125	0	25	0
76	76	Lakkalikööri	23	1	500 ml	18.00	57	0	20	0
77	77	Original Frankfurter grüne Soße	12	2	12 boxes	13.00	32	0	15	0

Dit getal kunnen we direct opvragen met de Query:

**SELECT MAX(UnitsInStock) FROM PRODUCTS;**

(No column name)	
1	125

Met deze uitkomst kunnen we alle gegevens opvragen van het product met dit maximale aantal units in voorraad in het magazijn:

**SELECT \* FROM PRODUCTS WHERE UnitsInStock = 125;**

	ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel	Discontinued
1	75	Rhönbräu Klosterbier	12	1	24 - 0.5 l bottles	7,75	125	0	25	0

In bovenstaand voorbeeld hebben we de gegevens van het product met het maximale aantal units op voorraad in het magazijn kunnen vinden in de volgende twee stappen:

1. Zoek welk product het maximale aantal units in voorraad in het magazijn heeft
2. Zoek alle gegevens van dit product

Bovenstaande resultaat hadden we echter ook in één stap kunnen vinden door het gebruik van een SUB-query:

**USE NORTHWIND;**

**SELECT \* FROM PRODUCTS WHERE UnitsInStock = (SELECT MAX(UnitsInStock) FROM PRODUCTS);**

	ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel	Discontinued
1	75	Rhönbräu Klosterbier	12	1	24 - 0.5 l bottles	7,75	125	0	25	0

De SUB query is het deel dat zich bevindt tussen de haken (). Het RDBMS rekent eerst de waarde uit van de SUB-query tussen de haken en geeft het resultaat ervan (in dit geval het getal 125) door aan de query buiten de haken.

Op dezelfde manier kunnen we ook alle gegevens opvragen van alle producten die een minimale aantal units op voorraad in het magazijn hebben:

**SELECT \* FROM PRODUCTS WHERE UnitsInStock = (SELECT MIN(UnitsInStock) FROM PRODUCTS);**

	ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel	Discontinued
1	5	Chef Anton's Gumbo Mix	2	2	36 boxes	21,35	0	0	0	1
2	17	Alice Mutton	7	6	20 - 1 kg tins	39,00	0	0	0	1
3	29	Thüringer Rostbratwurst	12	6	50 bags x 30 sausgs.	123,79	0	0	0	1
4	31	Gorgonzola Telino	14	4	12 - 100 g pkgs	12,50	0	70	20	0
5	53	Perth Pasties	24	6	48 pieces	32,80	0	0	0	1

Er blijken dus meerdere producten te zijn waarvan het aantal units op voorraad in het magazijn 0 is.

Op eenzelfde manier kunnen we ook zoeken naar de gegevens van alle medewerkers die rapporteren aan de directeur (medewerkers die direct onder de directeur vallen):

**SELECT \* FROM EMPLOYEES;**

EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	BirthDate	HireDate	Address	City	Region	PostalCode	Country	HomePhone	Extension	Photo	Notes	ReportsTo	PhotoPath
1	Davolio	Nancy	Sales Representative	Ms.	1981-02-09 00:00:00.000	1992-05-01 00:00:00.000	507 - 20th Ave. E. Apt. 2A	Seattle	WA	98122	USA	(206) 555-8987	5467	0x151C2F...	Education...	2	<a href="http://acoweb...">http://acoweb...</a>
2	Fetter	Andrew	Vice President, Sales	Dr.	1952-02-19 00:00:00.000	1982-08-14 00:00:00.000	908 W. Capital Way	Tacoma	WA	98401	USA	(206) 555-8402	3457	0x151C2F...	Andrew re...	NULL	<a href="http://acoweb...">http://acoweb...</a>
3	Leveleing	Janet	Sales Representative	Ms.	1963-08-30 00:00:00.000	1992-04-01 00:00:00.000	722 Main Bay Blvd.	Kirkland	WA	98033	USA	(206) 555-3412	3355	0x151C2F...	Janet has ...	2	<a href="http://acoweb...">http://acoweb...</a>
4	Pescod	Margaret	Sales Representative	Ms.	1973-08-19 00:00:00.000	1993-05-01 00:00:00.000	4110 Old Redmond Rd.	Redmond	WA	98052	USA	(206) 555-8122	5176	0x151C2F...	Margaret ...	2	<a href="http://acoweb...">http://acoweb...</a>
5	Buchanan	Steven	Sales Manager	Mr.	1955-03-04 00:00:00.000	1989-10-17 00:00:00.000	14 Corte Madera Dr.	London	NULL	SW1 8JR	UK	(71) 555-4848	3453	0x151C2F...	Steven B...	2	<a href="http://acoweb...">http://acoweb...</a>
6	Guyana	Michael	Sales Representative	Ms.	1963-07-02 00:00:00.000	1993-10-17 00:00:00.000	201 Cormier Street	London	NULL	EC2 7JR	UK	(71) 555-7773	428	0x151C2F...	Michael is ...	5	<a href="http://acoweb...">http://acoweb...</a>
7	Kings	Robert	Sales Representative	Mr.	1965-05-29 00:00:00.000	1994-01-02 00:00:00.000	Edgarhe Hollow, Winchester Way	London	NULL	RG1 9SP	UK	(71) 555-5589	465	0x151C2F...	Robert K...	5	<a href="http://acoweb...">http://acoweb...</a>
8	Callahan	Laura	Inside Sales Coordinator	Ms.	1958-01-09 00:00:00.000	1994-03-05 00:00:00.000	4723 - 11th Ave. N.E.	Seattle	WA	98105	USA	(206) 555-1189	2344	0x151C2F...	Laura rec...	2	<a href="http://acoweb...">http://acoweb...</a>
9	Dodsworth	Anne	Sales Representative	Ms.	1968-01-27 00:00:00.000	1994-11-15 00:00:00.000	7 Houndsworth Rd.	London	NULL	WG2 7LT	UK	(71) 555-4444	452	0x151C2F...	Anne has ...	5	<a href="http://acoweb...">http://acoweb...</a>

**SELECT EmployeeID, TitleOfCourtesy LastName, FirstName, ReportsTo FROM EMPLOYEES;**

	EmployeeID	LastName	FirstName	ReportsTo
1	1	Ms.	Nancy	2
2	2	Dr.	Andrew	NULL
3	3	Ms.	Janet	2
4	4	Mrs.	Margaret	2
5	5	Mr.	Steven	2
6	6	Mr.	Michael	5
7	7	Mr.	Robert	5
8	8	Ms.	Laura	2
9	9	Ms.	Anne	5

**SELECT EmployeeID FROM Employees WHERE ReportsTo IS NULL;**

	EmployeeID
1	2

**SELECT EmployeeID, TitleOfCourtesy LastName, FirstName, ReportsTo FROM EMPLOYEES WHERE ReportsTo = 2;**

	EmployeeID	LastName	FirstName	ReportsTo
1	1	Ms.	Nancy	2
2	3	Ms.	Janet	2
3	4	Mrs.	Margaret	2
4	5	Mr.	Steven	2
5	8	Ms.	Laura	2

Deze actie bestond weer uit de volgende twee stappen:

1. Selecteer het EmployeeID nummer van de Director (die geen rapportage hoeft te doen aan een andere medewerker)
2. Selecteer de gegevens van de medewerkers die rapporteren aan de Director met het EmployeeID 2

Dit kunnen we in een SUB-query weer samenvoegen in:

**SELECT EmployeeID, TitleOfCourtesy LastName, FirstName, ReportsTo FROM EMPLOYEES WHERE ReportsTo = (SELECT EmployeeID FROM Employees WHERE ReportsTo IS NULL);**

	EmployeeID	LastName	FirstName	ReportsTo
1	1	Ms.	Nancy	2
2	3	Ms.	Janet	2
3	4	Mrs.	Margaret	2
4	5	Mr.	Steven	2
5	8	Ms.	Laura	2

We kunnen ook SUB-queries over meerdere tabellen gebruiken. Stel bijvoorbeeld dat we in de database Northwind een overzicht met de adresgegevens willen hebben van alle klanten die in het verleden een order hebben geplaatst bij de verkoper Michael Suyama. Dan kunnen we dat in drie stappen op drie verschillende tabellen uitvoeren:

Stap1:

```
SELECT EmployeeID FROM Employees WHERE LastName = 'Suyama';
```

	EmployeeID
1	6

Stap2:

```
SELECT DISTINCT CustomerID FROM Orders WHERE EmployeeID = 6;
```

CustomerID
ALFKI
AROUT
BLAUS
BLONP
BOTTM
BSBEV
CHOPS
ERNSH
FOLIG
FOLKO
FRAN
GOURL
GREAL
HILAA
HUNGO
ISLAT
LACOR
LAMAI
LETSS
LILAS

Query executed successfully. MON-4GND6M3 (15.0 RTM) | USR\NIUP02 (67) | Northwind | 00:00:00 | 43 rows

Stap3:

```
SELECT CustomerID, CompanyName, ContactName, ContactTitle, Address, City, Region,
PostalCode FROM Customers WHERE CustomerID IN
('ALFKI','AROUT','BLAUS','BLONP','BOTTM','BSBEV','CHOPS','ERNSH','FOLIG','FOLKO','FRAN
K','GOURL','GREAL','HILAA','HUNGO','ISLAT','LACOR','LAMAI','LETSS','LILAS','LONEP','MA
GAA','MEREP','OLDWO','OTTIK','PICCO','QUEDE','QUEEN','RANCH','RATTC','SAVEA','SEVES','
SPECD','SPLIR','SUPRD','THEBI','TOMSP','TRAIH','VAFFE','VINET','WANDK','WARTH','WOLZA
');
```

CustomerID	CompanyName	ContactName	ContactTitle	Address	City	Region	PostalCode
1 ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative	Obere Str. 57	Berlin	NULL	12209
2 AROUT	Around the Horn	Thomas Hardy	Sales Representative	120 Hanover Sq.	London	WA1 1DP	
3 BLAUS	Blauer See Delikatessen	Hanna Moos	Sales Representative	Forsterstr. 57	Mannheim	NULL	68306
4 BLONP	Blondesdssl père et fils	Frédérique Ceteau	Marketing Manager	24, place Kléber	Strasbourg	NULL	67000
5 BOTTM	Botton-Dollar Markets	Elizabeth Lincoln	Accounting Manager	23 Tsawassen Blvd.	Tsawassen	BC	T2F 8M4
6 BSBEV	B's Beverages	Victoria Ashworth	Sales Representative	Fauntleroy Circus	London	NULL	EC2 5NT
7 CHOPS	Chop-suey Chinese	Yang Wang	Owner	Hauptstr. 29	Bern	NULL	3012
8 ERNSH	Ernst Handel	Roland Mendel	Sales Manager	Kirchgasse 6	Graz	NULL	8010
9 FOLIG	Folies gourmandes	Marine Ramé	Assistant Sales Agent	164, chaussee de Tournai	Lille	NULL	59000
10 FOLKO	Folk och få HB	Maria Larsson	Owner	Åkersgatan 24	Bräcke	NULL	S-844 67
11 FRAN	Frankenversand	Peter Franken	Marketing Manager	Berliner Platz 43	München	NULL	80805
12 GOURL	Gourmet Lanchonetes	André Fonseca	Sales Associate	Av. Brasil, 442	Campinas	SP	04376-786
13 GREAL	Great Lakes Food Market	Howard Snyder	Marketing Manager	2732 Baker Blvd.	Eugene	OR	97403
14 HILAA	HILARION-Abastos	Carlos Hernández	Sales Representative	Camara 22 con Ave. Carlos Soublette #8-35	San Cristóbal	Táchira	5022
15 HUNGO	Hungry Owl All-Night Grocers	Patricia McKenna	Sales Associate	8 Johnstown Road	Cork	Co. Cork	NULL
16 ISLAT	Island Trading	Helen Bennett	Marketing Manager	Garden House Crother Way	Cowes	Isle of Wight	PO31 7PJ
17 LACOR	La corne d'abondance	Daniel Tonini	Sales Representative	67, avenue de l'Europe	Versailles	NULL	78000
18 LAMAI	La maison d'Asie	Annette Roulet	Sales Manager	1 rue Alsace-Lorraine	Toulouse	NULL	31000
19 LETSS	Let's Stop N Shop	Jaimie Yorke	Owner	87 Polk St. Suite 5	San Francisco	CA	94117
20 LILAS	LILA-Supermercado	Carlos González	Accounting Manager	Carrera 52 con Ave. Bolívar #65-98 Llano Largo	Barquisimeto	Lara	3508

Query executed successfully. MON-4GND6M3 (15.0 RTM) | USR\NIUP02 (67) | Northwind | 00:00:00 | 43 rows

Ook deze actie in drie stappen hadden we in één actie kunnen uitvoeren m.b.v. SUB-queries waarbij het resultaat van de SUB-query telkens gebruikt wordt in de volgende query:

```
SELECT CustomerID, CompanyName, ContactName, ContactTitle, Address, City, Region,
PostalCode FROM Customers WHERE CustomerID IN
(SELECT DISTINCT CustomerID FROM Orders WHERE EmployeeID =
(SELECT EmployeeID FROM Employees WHERE LastName = 'Suyama'))
```

);

CustomerID	CompanyName	ContactName	ContactTitle	Address	City	Region	PostalCode
1	ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative	Oberseestr. 57	Berlin	NULL
2	AROUT	Around the Horn	Thomas Hardy	Sales Representative	120 Hanover Sq.	London	NULL
3	BLAUS	Blauer See Delikatessen	Hanna Moos	Sales Representative	Foersterstr. 57	Mannheim	NULL
4	BLONP	Blondesdal père et fils	Frédérique Citeaux	Marketing Manager	24, place Kléber	Strasbourg	NULL
5	BOTTM	Bottom-Dollar Markets	Elizabeth Lincoln	Accounting Manager	23 Tsawassen Blvd.	Tsawassen	BC T2F 8M4
6	BSBEV	B's Beverages	Victoria Ashworth	Sales Representative	Faunoroy Circus	London	NULL
7	CHOPS	Chop-suey Chinese	Yang Wang	Owner	Hauptstr. 29	Bern	NULL
8	ERNSH	Ernst Handel	Roland Mendel	Sales Manager	Kirchgasse 6	Graz	NULL
9	FOLIG	Folies gourmandes	Martine Ranöd	Assistant Sales Agent	184, chaussée de Tourmai	Lille	NULL
10	FOLKO	Fok och fä HB	Maria Larsson	Owner	Åkersgatan 24	Bräcke	NULL
11	FRANK	Frankenversand	Peter Franken	Marketing Manager	Berliner Platz 43	München	NULL
12	GOURL	Gourmet Lanchonetes	André Fonseca	Sales Associate	Av. Brasil, 442	Campinas	SP 04676-786
13	GREAL	Great Lakes Food Market	Howard Snyder	Marketing Manager	2732 Baker Blvd.	Eugene	OR 97403
14	HILAA	HILARIONAbastos	Carlos Hernández	Sales Representative	Carrera 22 con Ave. Carlos Soublette #8-35	San Cristóbal	Táchira 5022
15	HUNGO	Hungry Owl All-Night Grocers	Patricia McKenna	Sales Associate	8 Johnstown Road	Cork	Co. Cork
16	ISLAT	Island Trading	Helen Bennett	Marketing Manager	Garden House Crowther Way	Cowes	Isle of Wight PO31 7PJ
17	LACOR	La corne d'abondance	Daniel Tonini	Sales Representative	67, avenue de l'Europe	Versailles	NULL
18	LAMAI	La maison d'Asie	Annette Roulet	Sales Manager	1 rue Alsace-Lorraine	Toulouse	NULL
19	LETSS	Le's Shop N Shop	Jáime Yorres	Owner	87 Polk St. Suite 5	San Francisco	CA 94117
20	LILAS	LILA-Supermercado	Carlos González	Accounting Manager	Carrera 52 con Ave. Bolívar #65-98 Llano Largo	Barquisimeto	Lara 3508

Query executed successfully.

MON-4GND6N3 (15.0 RTM) | USR\NIJP02 (67) | Northwind | 00:00:00 | 43 rows

Deze SUB-query in een SUB-query noemen we **geneste SUB-queries**. In dit laatste voorbeeld bestaat de query uit drie geneste SUB-queries. Het is technisch mogelijk om zeer veel SUB-queries te nesten, maar het wordt daarmee wel steeds abstracter, complexer en moeilijker te begrijpen.

## Hoofdstuk 14 VIEWS

In een tabel worden rijen met gegevens werkelijk fysiek opgeslagen op disk. Een tabel neemt dan ook een bepaalde hoeveelheid ruimte in. Hoe meer rijen, des te meer opslagruimte.

Views zijn tabellen die wel zichtbaar zijn voor gebruikers, maar die geen opslagruimte innemen. Ze worden daarom ook wel virtuele of afgeleide tabellen genoemd. Een view is een tabel zonder werkelijke rijen, maar een die zich wel gedraagt alsof die er wel zijn. Een view kan worden gezien als een voorschrift of formule om bepaalde gegevens uit de basistabellen in een virtuele tabel samen te voegen. We gebruiken het woord virtueel omdat de inhoud van een view alleen bestaat als de view in een instructie gebruikt wordt. Op zo'n moment voert SQL het in de viewformule opgenomen voorschrift uit en lijkt het voor de gebruiker net alsof er met een echte tabel gewerkt wordt.

### 14.1 Views

Als we alle gegevens uit de tabel Employees selecteren dan zien we alle gegevens van alle medewerkers:

**USE NORTHWIND;**

**SELECT \* FROM Employees;**

EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	BirthDate	HireDate	Address	City	Region	PostalCode	Country	HomePhone	Extension	Photo	Notes	ReportsTo	PhotoPath
1 1	Davolio	Nancy	Sales Representative	Ms.	1984-12-08 00:00:00.000	1992-05-01 00:00:00.000	507 - 20th Ave. E. Apt. 2A	Seattle	WA	98122	USA	(206) 555-8857	5467	0x151C2F000...	Educatio...	2	http://acow...
2 2	Fuller	Andrew	Vice President, Sales	Dr.	1952-02-19 00:00:00.000	1992-05-14 00:00:00.000	908 W. Capital Way	Tacoma	WA	98401	USA	(206) 555-8482	3457	0x151C2F000...	Andrew r...	NULL	http://acow...
3 3	Leverling	Janet	Sales Representative	Ms.	1963-08-30 00:00:00.000	1992-04-01 00:00:00.000	722 Mass Bay Blvd.	Kirkland	WA	98033	USA	(206) 555-3412	3355	0x151C2F000...	Janet ha...	2	http://acow...
4 4	Peacock	Margaret	Sales Representative	Mrs.	1937-09-19 00:00:00.000	1993-05-03 00:00:00.000	4110 Old Redmond Rd.	Redmond	WA	98052	USA	(206) 555-8122	5176	0x151C2F000...	Margaret...	2	http://acow...
5 5	Buchanan	Steven	Sales Manager	Mr.	1955-03-04 00:00:00.000	1993-10-17 00:00:00.000	14 Garrett Hill	London	NULL	SW1 8JR	UK	(71) 555-4848	3453	0x151C2F000...	Steven B...	2	http://acow...
6 6	Suyama	Michael	Sales Representative	Mr.	1963-07-02 00:00:00.000	1993-10-17 00:00:00.000	Coventry House, Miner Rd.	London	NULL	EC2 7JR	UK	(71) 555-7773	428	0x151C2F000...	Michael i...	5	http://acow...
7 7	King	Robert	Sales Representative	Mr.	1960-05-29 00:00:00.000	1994-01-02 00:00:00.000	Edgemoor Hollow, Winchester Way	London	NULL	RG1 9SP	UK	(71) 555-5598	465	0x151C2F000...	Robert Ki...	5	http://acow...
8 8	Callahan	Laura	Inside Sales Coordinator	Ms.	1958-01-09 00:00:00.000	1994-03-05 00:00:00.000	4726 - 11th Ave. N.E.	Seattle	WA	98105	USA	(206) 555-1189	2344	0x151C2F000...	Laura re...	2	http://acow...
9 9	Dodsworth	Anne	Sales Representative	Ms.	1966-01-27 00:00:00.000	1994-11-15 00:00:00.000	7 Houndstooth Rd.	London	NULL	WG2 7LT	UK	(71) 555-4444	452	0x151C2F000...	Anne has...	5	http://acow...

Stel dat we de gegevens van de directeur niet zichtbaar willen maken, dan kunnen we een view maken waarin de gegevens van de directeur ontbreken:

**CREATE VIEW EmployeesWithoutDirector AS  
SELECT \* FROM Employees WHERE ReportsTo IS NOT NULL;**

**SELECT \* FROM EmployeesWithoutDirector;**

EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	BirthDate	HireDate	Address	City	Region	PostalCode	Country	HomePhone	Extension	Photo	Notes	ReportsTo	PhotoPath
1 1	Davolio	Nancy	Sales Representative	Ms.	1984-12-08 00:00:00.000	1992-05-01 00:00:00.000	507 - 20th Ave. E. Apt. 2A	Seattle	WA	98122	USA	(206) 555-8857	5467	0x151C2F000...	Educatio...	2	http://acow...
2 3	Leverling	Janet	Sales Representative	Ms.	1963-08-30 00:00:00.000	1992-04-01 00:00:00.000	722 Mass Bay Blvd.	Kirkland	WA	98033	USA	(206) 555-3412	3355	0x151C2F000...	Janet ha...	2	http://acow...
3 4	Peacock	Margaret	Sales Representative	Mrs.	1937-09-19 00:00:00.000	1993-05-03 00:00:00.000	4110 Old Redmond Rd.	Redmond	WA	98052	USA	(206) 555-8122	5176	0x151C2F000...	Margaret...	2	http://acow...
4 5	Buchanan	Steven	Sales Manager	Mr.	1955-03-04 00:00:00.000	1993-10-17 00:00:00.000	14 Garrett Hill	London	NULL	SW1 8JR	UK	(71) 555-4848	3453	0x151C2F000...	Steven B...	2	http://acow...
5 6	Suyama	Michael	Sales Representative	Mr.	1963-07-02 00:00:00.000	1993-10-17 00:00:00.000	Coventry House, Miner Rd.	London	NULL	EC2 7JR	UK	(71) 555-7773	428	0x151C2F000...	Michael i...	5	http://acow...
6 7	King	Robert	Sales Representative	Mr.	1960-05-29 00:00:00.000	1994-01-02 00:00:00.000	Edgemoor Hollow, Winchester Way	London	NULL	RG1 9SP	UK	(71) 555-5598	465	0x151C2F000...	Robert Ki...	5	http://acow...
7 8	Callahan	Laura	Inside Sales Coordinator	Ms.	1958-01-09 00:00:00.000	1994-03-05 00:00:00.000	4726 - 11th Ave. N.E.	Seattle	WA	98105	USA	(206) 555-1189	2344	0x151C2F000...	Laura re...	2	http://acow...
8 9	Dodsworth	Anne	Sales Representative	Ms.	1966-01-27 00:00:00.000	1994-11-15 00:00:00.000	7 Houndstooth Rd.	London	NULL	WG2 7LT	UK	(71) 555-4444	452	0x151C2F000...	Anne has...	5	http://acow...

Stel dat we de adresgegevens van de medewerkers niet zichtbaar willen maken, dan kunnen we een view maken waarin de adresgegevens ontbreken:

**CREATE VIEW EmployeesWithoutAddress AS  
SELECT EmployeeID, LastName, FirstName, TitleOfCourtesy FROM Employees;**

**SELECT \* FROM EmployeesWithoutAddress;**

	EmployeeID	LastName	FirstName	TitleOfCourtesy
1	1	Davolio	Nancy	Ms.
2	2	Fuller	Andrew	Dr.
3	3	Leverling	Janet	Ms.
4	4	Peacock	Margaret	Mrs.
5	5	Buchanan	Steven	Mr.
6	6	Suyama	Michael	Mr.
7	7	King	Robert	Mr.
8	8	Callahan	Laura	Ms.
9	9	Dodsworth	Anne	Ms.

We kunnen ook een view maken over meerdere tabellen.

Stel dat we de bedrijfsnaam en de contactpersoon van de klanten uit de Customers tabel samen met de Ordernummers en Orderdata uit de Orders tabel in één overzicht willen hebben dan kunnen we dat doen door een query uit te voeren op de twee tabellen:

```
SELECT Customers.CompanyName, Customers.ContactName, Orders.OrderID, Orders.OrderDate
FROM Customers, Orders
WHERE Customers.CustomerID = Orders.CustomerID;
```

	CompanyName	ContactName	OrderID	OrderDate
1	Vins et alcools Chevalier	Paul Henrot	10248	1996-07-04 00:00:00.000
2	Toms Spezialitäten	Karin Josephs	10249	1996-07-05 00:00:00.000
3	Hanai Carnes	Mario Pontes	10250	1996-07-08 00:00:00.000
4	Victuailles en stock	Mary Saviley	10251	1996-07-08 00:00:00.000
5	Suprêmes délices	Pascale Carréain	10252	1996-07-09 00:00:00.000
6	Hanai Carnes	Mario Pontes	10253	1996-07-10 00:00:00.000
7	Chop-suey Chinese	Yang Wang	10254	1996-07-11 00:00:00.000
8	Richter Supermarkt	Michael Holz	10255	1996-07-12 00:00:00.000
9	Wellington Importadora	Paula Parente	10256	1996-07-15 00:00:00.000
10	HILARION-Abastos	Carlos Hernández	10257	1996-07-16 00:00:00.000
11	Ernst Handel	Roland Mendel	10258	1996-07-17 00:00:00.000
12	Centro comercial Moctezuma	Francisco Chang	10259	1996-07-18 00:00:00.000
13	Ottiles Käseladen	Hennette Pfalzheim	10260	1996-07-19 00:00:00.000
14	Que Delicia	Bernardo Batista	10261	1996-07-19 00:00:00.000
15	Rattlesnake Canyon Grocery	Paula Wilson	10262	1996-07-22 00:00:00.000
16	Ernst Handel	Roland Mendel	10263	1996-07-23 00:00:00.000
17	Folk och få HB	Maria Larsson	10264	1996-07-24 00:00:00.000
18	Blondesddsl père et fils	Frédérique Creux	10265	1996-07-25 00:00:00.000
19	Wartian Herku	Pirkko Koskitalo	10266	1996-07-26 00:00:00.000
20	Frankenversand	Peter Franken	10267	1996-07-29 00:00:00.000

✓ Query executed successfully. | MON-4GND6M3 (15.0 RTM) | USR\NIJP02 (67) | Northwind | 00:00:00 | 830 rows

Echter is het voor een gebruiker lastig om telkens de query in te typen. We kunnen daarom een view maken die deze query voor ons uitvoert:

```
CREATE VIEW CustomersOrders AS
SELECT Customers.CompanyName, Customers.ContactName, Orders.OrderID, Orders.OrderDate
FROM Customers, Orders
WHERE Customers.CustomerID = Orders.CustomerID;
```

De gebruiker kan nu direct de view gebruiken in plaats van de query alsof de view een normale tabel is:

```
SELECT * FROM CustomersOrders;
```

	CompanyName	ContactName	OrderID	OrderDate
1	Vins et alcools Chevalier	Paul Hennot	10248	1996-07-04 00:00:00.000
2	Toms Spezialitäten	Karin Josepha	10249	1996-07-05 00:00:00.000
3	Hanai Carnes	Mario Pontes	10250	1996-07-08 00:00:00.000
4	Victuailles en stock	Mary Saveley	10251	1996-07-08 00:00:00.000
5	Suprêmes délices	Pascale Cartain	10252	1996-07-09 00:00:00.000
6	Hanai Carnes	Mario Pontes	10253	1996-07-10 00:00:00.000
7	Chop-suey Chinese	Yang Wang	10254	1996-07-11 00:00:00.000
8	Richter Supermarkt	Michael Holz	10255	1996-07-12 00:00:00.000
9	Wellington Importadora	Paula Parente	10256	1996-07-15 00:00:00.000
10	HILARION-Abastos	Carlos Hernández	10257	1996-07-16 00:00:00.000
11	Ernst Handel	Roland Mendel	10258	1996-07-17 00:00:00.000
12	Centro comercial Moctezuma	Franisco Chang	10259	1996-07-18 00:00:00.000
13	Ottilies Käseladen	Henriette Pfalzheim	10260	1996-07-19 00:00:00.000
14	Que Delicia	Bernardo Batista	10261	1996-07-19 00:00:00.000
15	Rattlesnake Canyon Grocery	Paula Wilson	10262	1996-07-22 00:00:00.000
16	Ernst Handel	Roland Mendel	10263	1996-07-23 00:00:00.000
17	Folk och fa HB	Maria Larsson	10264	1996-07-24 00:00:00.000
18	Blondedds l père et fils	Frédérique Citeaux	10265	1996-07-25 00:00:00.000
19	Wartian Herku	Pirkko Koskitalo	10266	1996-07-26 00:00:00.000
20	Frankenversand	Peter Franken	10267	1996-07-29 00:00:00.000

✓ Query executed successfully.

MON-4GND6M3 (15.0 RTM) | USR\NIJP02 (67) | Northwind | 00:00:00 | 830 rows

Een view lijkt voor de gebruiker dus op een tabel maar is in werkelijkheid een opgeslagen SQL query die uitgevoerd wordt op het moment dat deze gebruikt wordt.

## Hoofdstuk 15 JOINS

### 15.1 Joins

We kunnen gegevens van twee of meerdere tabellen samenvoegen tot één tabel als resultaat alsof de gegevens uit één grote tabel is gekomen. Wanneer we gegevens van verschillende tabellen combineren tot één tabel spreken we over een **Join** van tabellen. De kolommen waarop de join wordt uitgevoerd worden **join-kolommen** genoemd.

### 15.2 Impliciete en Expliciete Joins

We onderscheiden twee soorten joins:

- **Impliciete joins**

Impliciete joins zijn joins waarbij het woord JOIN verstopt zit in de SELECT-instructie. Een join wordt dan gevormd door enkel specificaties uit de FROM-component samen met enkele condities in de WHERE-component. De conditie in de WHERE-component waarmee de kolomwaarden van de tabellen met elkaar worden vergeleken heet de **join conditie**.

De SQL syntax voor een impliciete join tussen twee tabellen is:

```
SELECT <tabel1.kolom1>, <tabel2.kolom2>,...  
FROM <tabel1, tabel2>  
WHERE <tabel1.kolomA = tabel2.kolomB>  
AND...;
```

- **Expliciete joins**

Expliciete joins zijn joins waar het woord JOIN expliciet staat vermeld in de SELECT-instructie.

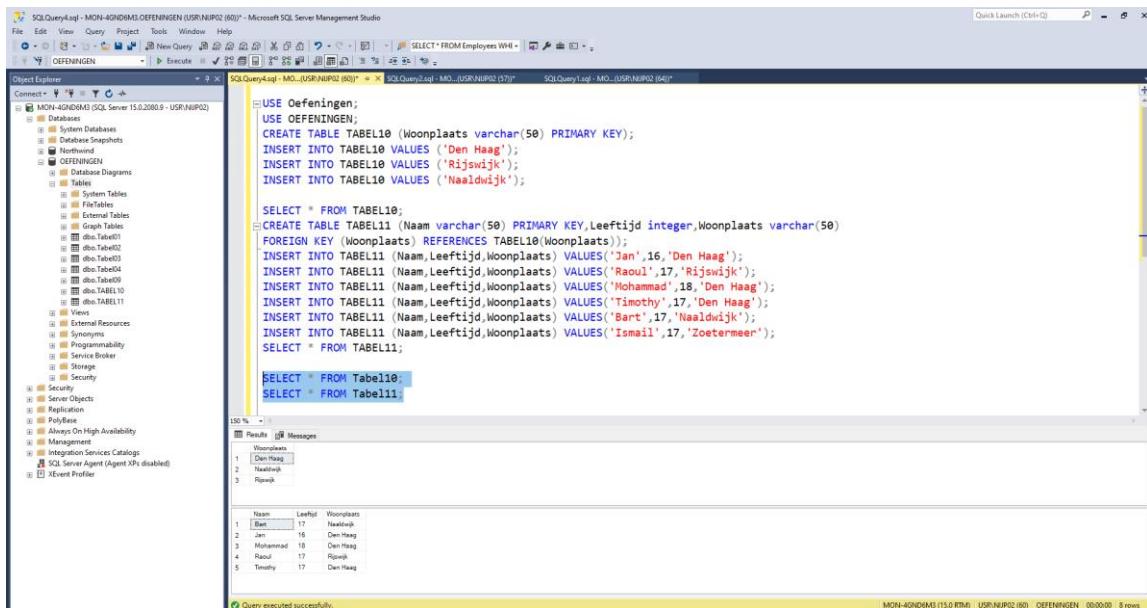
De SQL syntax voor een expliciete join tussen twee tabellen is:

```
SELECT <tabel1.kolom1>, <tabel2.kolom2>,...  
FROM <tabel1> [INNER|OUTER|CROSS] JOIN <tabel2>  
ON <tabel1.kolomA = tabel2.kolomB>  
WHERE...;
```

Een Join is niet beperkt tot twee tabellen. Een FROM-component mag meerdere tabellen bevatten.

De meeste Join-condities die geformuleerd worden, vergelijken sleutelkolommen met elkaar. Dit is echter geen vereiste. Het is bijvoorbeeld ook mogelijk om een datum en een jaartal met elkaar te vergelijken die beiden geen sleutelkolommen zijn.

Als voorbeeld voor een impliciete join nemen we de tabellen Tabel10 en Tabel11 die we hebben gemaakt bij de uitleg van de Foreign Key constraint:



The screenshot shows the Microsoft SQL Server Management Studio interface. On the left, the Object Explorer displays the database structure of 'OEFENINGEN'. Two queries are run in separate windows:

```

USE OEFENINGEN;
CREATE TABLE TABEL10 (Woonplaats varchar(50) PRIMARY KEY);
INSERT INTO TABEL10 VALUES ('Den Haag');
INSERT INTO TABEL10 VALUES ('Rijswijk');
INSERT INTO TABEL10 VALUES ('Naaldwijk');

SELECT * FROM TABEL10;

USE OEFENINGEN;
CREATE TABLE TABEL11 (Naam varchar(50) PRIMARY KEY, Leeftijd integer, Woonplaats varchar(50));
FOREIGN KEY (Woonplaats) REFERENCES TABEL10(Woonplaats);
INSERT INTO TABEL11 (Naam, Leeftijd, Woonplaats) VALUES ('Jan', 16, 'Den Haag');
INSERT INTO TABEL11 (Naam, Leeftijd, Woonplaats) VALUES ('Raoul', 17, 'Rijswijk');
INSERT INTO TABEL11 (Naam, Leeftijd, Woonplaats) VALUES ('Mohammad', 18, 'Den Haag');
INSERT INTO TABEL11 (Naam, Leeftijd, Woonplaats) VALUES ('Timothy', 17, 'Den Haag');
INSERT INTO TABEL11 (Naam, Leeftijd, Woonplaats) VALUES ('Bart', 17, 'Naaldwijk');
INSERT INTO TABEL11 (Naam, Leeftijd, Woonplaats) VALUES ('Ismail', 17, 'Zoetermeer');

SELECT * FROM TABEL11;
  
```

The 'Messages' pane shows the execution results:

	Woonplaats
1	Den Haag
2	Naaldwijk
3	Rijswijk

The 'Results' pane shows the combined data from both tables:

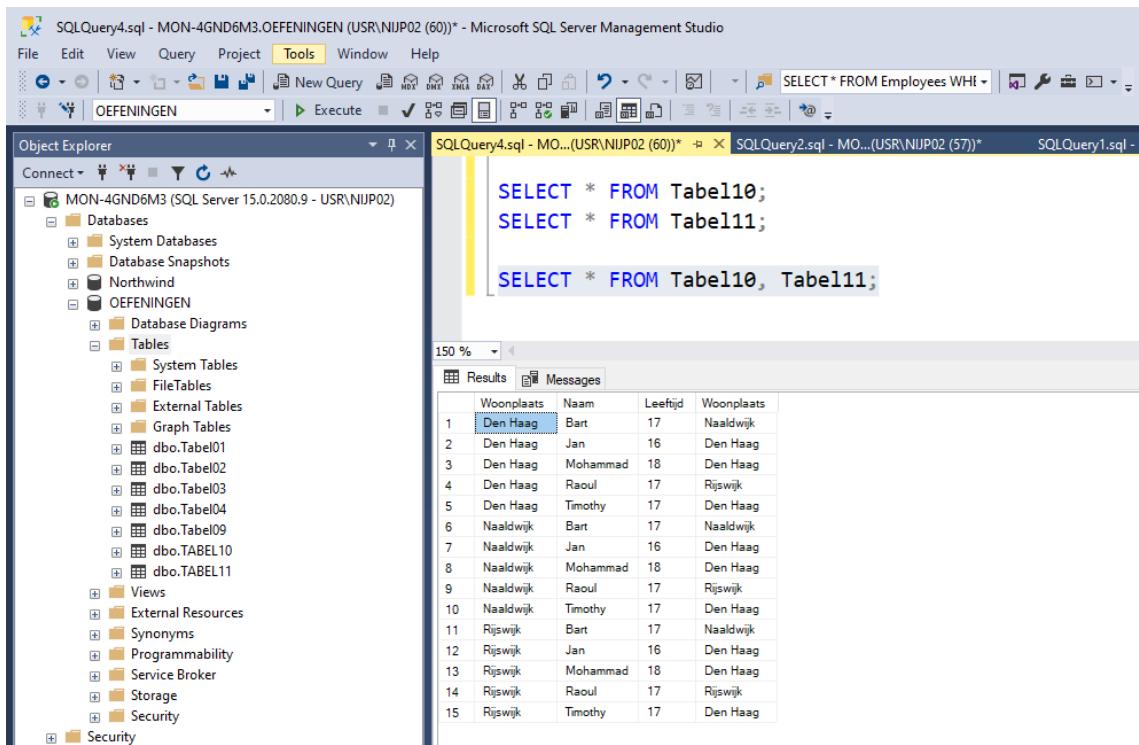
	Naam	Leeftijd	Woonplaats
1	Bart	17	Naaldwijk
2	Jan	16	Den Haag
3	Mohammad	18	Den Haag
4	Raoul	17	Rijswijk
5	Timothy	17	Den Haag
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			

A status bar at the bottom indicates: MON-4GND6M3 (15.0 RTM) USR\NIJP02 (80) OEFENINGEN 00:00:00 8 rows.

Tabel10 bevat 1 kolom met 3 rijen en Tabel11 bevat 3 kolommen met 5 rijen. Als we nu op de volgende manier gegevens uit Tabel10 en Tabel11 zouden samenvoegen in de volgende query:

`SELECT * FROM Tabel10, Tabel11;`

Dan krijgen we het volgende resultaat:



The screenshot shows the Microsoft SQL Server Management Studio interface. On the left, the Object Explorer displays the database structure of 'OEFENINGEN'. A single query window contains the following code:

```

SELECT * FROM Tabel10;
SELECT * FROM Tabel11;
  
```

The 'Results' pane shows the combined data from both tables:

	Woonplaats	Naam	Leeftijd	Woonplaats
1	Den Haag	Bart	17	Naaldwijk
2	Den Haag	Jan	16	Den Haag
3	Den Haag	Mohammad	18	Den Haag
4	Den Haag	Raoul	17	Rijswijk
5	Den Haag	Timothy	17	Den Haag
6	Naaldwijk	Bart	17	Naaldwijk
7	Naaldwijk	Jan	16	Den Haag
8	Naaldwijk	Mohammad	18	Den Haag
9	Naaldwijk	Raoul	17	Rijswijk
10	Naaldwijk	Timothy	17	Den Haag
11	Rijswijk	Bart	17	Naaldwijk
12	Rijswijk	Jan	16	Den Haag
13	Rijswijk	Mohammad	18	Den Haag
14	Rijswijk	Raoul	17	Rijswijk
15	Rijswijk	Timothy	17	Den Haag

Hoewel Tabel10 drie rijen bevat en Tabel11 vijf rijen geeft de query `SELECT * FROM Tabel10, Tabel11;` als resultaat 15 rijen. De query geeft dus het aantal rijen van Tabel10 vermenigvuldigd met het aantal rijen in Tabel11. Dit noemen we een **Cartesisch Product**. In het resultaat komt alle combinaties van de waarden van de eerste tabel gecombineerd met alle waarden van de tweede

tabel. In bovenstaand voorbeeld wordt elke woonplaats gecombineerd met elke persoon, of andersom ook elke persoon gecombineerd met elke woonplaats. Een Cartesisch Product is in het algemeen een nutteloos resultaat, maar wel belangrijk om te weten. Wanneer je in de praktijk een keer een onverwacht groot aantal rijen uit een query krijg, dan is het verstandig na te lopen of er niet een denkfout is gemaakt waardoor ergens in de query een cartesisch product is ontstaan.

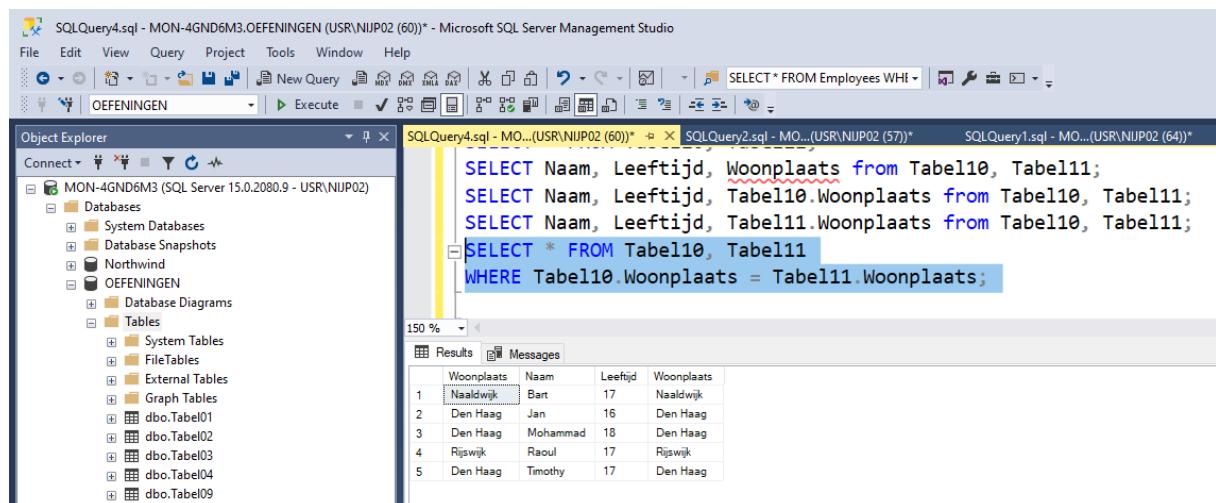
We kunnen een cartesisch product voorkomen door in de query binnen de WHERE clause (join conditie) aan te geven dat de koppeling tussen de tabellen Tabel10 en Tabel11 in de Foreign Key gedefinieerd is op de kolommen Woonplaats:

```
SELECT * FROM Tabel10, Tabel11
WHERE Tabel10.Woonplaats = Tabel11.Woonplaats;
```

Hierbij geven we explicet aan welke kolom we bedoelen door de naam van de tabel ervoor te zetten en tussen de tabelnaam en de kolomnaam een punt te zetten <tabelnaam>.<kolomnaam>.

Tabel10.Woonplaats betekent dus de kolom Woonplaats uit Tabel10 en Tabel11.Woonplaats betekent dus de kolom Woonplaats uit Tabel11.

Dit geeft het volgende resultaat:



	Woonplaats	Naam	Leeftijd	Woonplaats
1	Naaldwijk	Bart	17	Naaldwijk
2	Den Haag	Jan	16	Den Haag
3	Den Haag	Mohammad	18	Den Haag
4	Rijswijk	Raoul	17	Rijswijk
5	Den Haag	Timothy	17	Den Haag

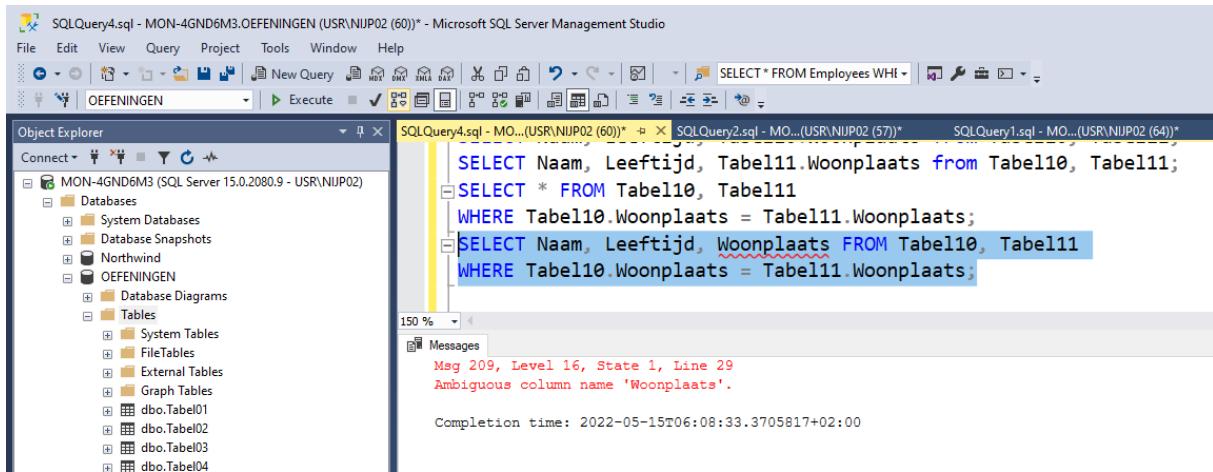
Door in de WHERE clause te stellen dat we alleen de waarden willen zien waarbij de waarde van de kolom Woonplaats in tabel Tabel10 gelijk is aan de kolom Woonplaats in de tabel Tabel11 hebben we dus het cartesische product voorkomen.

In het resultaat zien we echter nog wel 4 kolommen: Woonplaats, Naam, Leeftijd, Woonplaats. De meest linker kolom Woonplaats komt uit Tabel10 en de overige kolommen komen uit Tabel11. Er zit dus dubbele informatie in het resultaat. Om de dubbele gegevens uit het resultaat weg te laten zouden we kunnen proberen om in plaats van de \* de gewenste kolommen te benoemen:

```
SELECT Naam, Leeftijd, Woonplaats FROM Tabel10, Tabel11
WHERE Tabel10.Woonplaats = Tabel11.Woonplaats;
```

Dit geeft echter een foutmelding:

```
Msg 209, Level 16, State 1, Line 29
Ambiguous column name 'Woonplaats'.
Completion time: 2022-05-15T06:08:33.3705817+02:00
```



```

SELECT Naam, Leeftijd, Tabel11.Woonplaats from Tabel10, Tabel11;
SELECT * FROM Tabel10, Tabel11
WHERE Tabel10.Woonplaats = Tabel11.Woonplaats;
SELECT Naam, Leeftijd, Woonplaats FROM Tabel10, Tabel11
WHERE Tabel10.Woonplaats = Tabel11.Woonplaats;

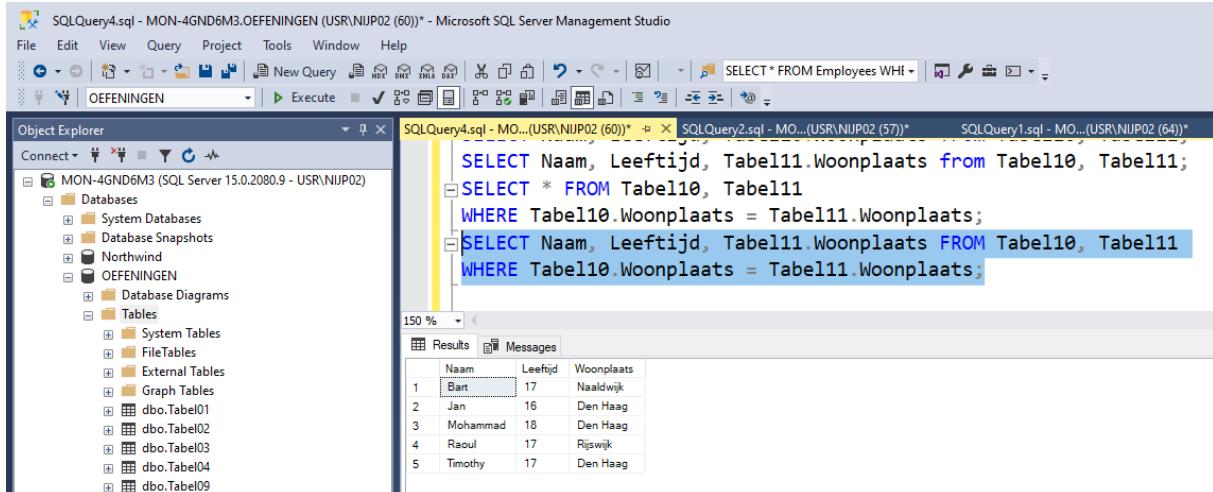
```

De foutmelding geeft aan dat de vraag om Woonplaats te ambitieus is, omdat SQL Server niet weet welke kolom Woonplaats je wilt hebben. Wil je de kolom Woonplaats uit Tabel10 of uit Tabel11 hebben? We moeten daarom expliciet aangeven welke kolom Woonplaats we willen hebben door voor de kolomnaam de tabelnaam te vermelden met daar tussen een punt <tabelnaam>.<kolomnaam>:

```

SELECT Naam, Leeftijd, Tabel11.Woonplaats FROM Tabel10, Tabel11
WHERE Tabel10.Woonplaats = Tabel11.Woonplaats;

```



```

SELECT Naam, Leeftijd, Tabel11.Woonplaats from Tabel10, Tabel11;
SELECT * FROM Tabel10, Tabel11
WHERE Tabel10.Woonplaats = Tabel11.Woonplaats;
SELECT Naam, Leeftijd, Tabel11.Woonplaats FROM Tabel10, Tabel11
WHERE Tabel10.Woonplaats = Tabel11.Woonplaats;

```

	Naam	Leeftijd	Woonplaats
1	Bart	17	Naaldwijk
2	Jan	16	Den Haag
3	Mohammed	18	Den Haag
4	Raoul	17	Rijswijk
5	Timothy	17	Den Haag

Dus als een kolomnaam gebruikt wordt die voorkomt in meer dan één van de tabellen die in de FROM-component gespecificeerd zijn, is het verplicht een tabelspecificatie in de kolomspecificatie op te nemen.

In bovenstaand voorbeeld is het uiteindelijke resultaat van de query gelijk aan het resultaat van alle gegevens uit Tabel11. Maar indien Tabel10 meerdere kolommen zou hebben gehad die wij ook hadden willen selecteren dan zou een query op slechts Tabel11 niet voldoende zijn.

De volgorde binnen de WHERE clausule heeft géén invloed op het resultaat. De SELECT-component is de enige component waarin bepaald wordt in welke volgorde de kolommen komen te staan. De volgende twee instructies zijn dus wat betreft hun resultaten gelijkwaardig:

```

SELECT Naam, Leeftijd, Tabel11.Woonplaats FROM Tabel10, Tabel11

```

```
WHERE Tabel10.Woonplaats = Tabel11.Woonplaats;
```

En:

```
SELECT Naam, Leeftijd, Tabel11.Woonplaats FROM Tabel11, Tabel10
WHERE Tabel11.Woonplaats = Tabel10.Woonplaats;
```

### 15.3 Pseudoniemen

Als er meerdere tabelspecificaties in de FROM-component voorkomen, is het soms gemakkelijk zogenaamde **pseudoniemen** te gebruiken. Een andere naam die ook wel eens wordt gebruikt voor pseudoniem is **alias**. Pseudoniemen zijn tijdelijke, alternatieve namen voor tabellenamen, bv:

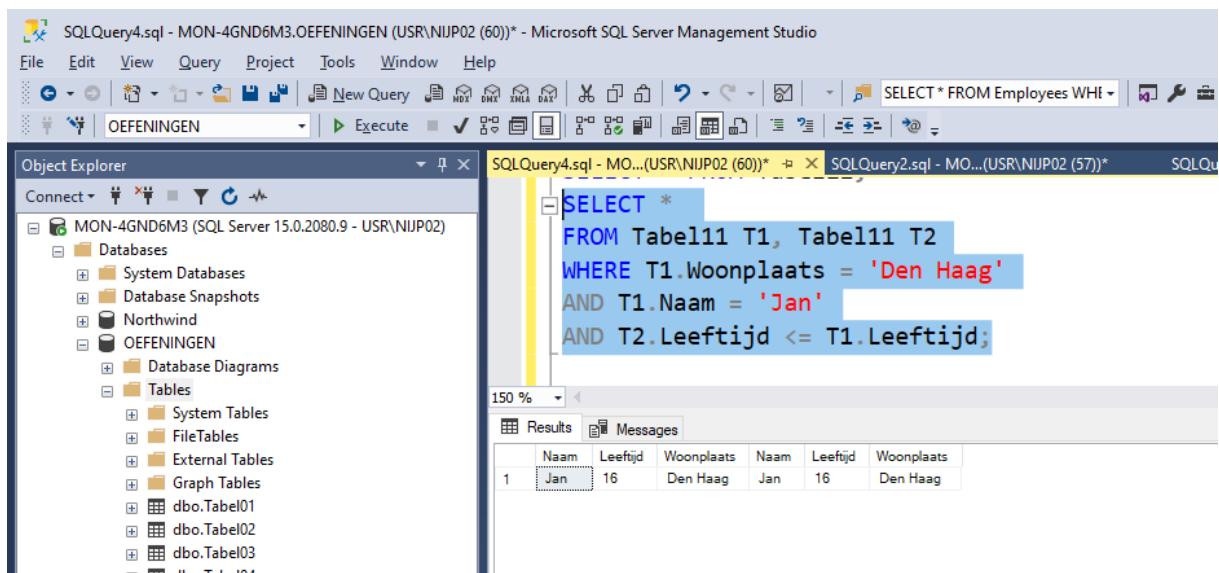
```
SELECT Naam, Leeftijd, T11.Woonplaats
FROM Tabel11 AS T11, Tabel10 AS T10
WHERE T10.Woonplaats = T11.Woonplaats;
```

Het woord AS mag eventueel ook weggelaten worden:

```
SELECT Naam, Leeftijd, T11.Woonplaats
FROM Tabel11 T11, Tabel10 T10
WHERE T10.Woonplaats = T11.Woonplaats;
```

Het is ook mogelijk om een join uit te voeren op 2x dezelfde tabel. In dat geval is het gebruik van pseudoniemen niet meer vrijblijvend maar verplicht, bijvoorbeeld:

```
SELECT *
FROM Tabel11 T1, Tabel11 T2
WHERE T1.Woonplaats = 'Den Haag'
AND T1.Naam = 'Jan'
AND T2.Leeftijd <= T1.Leeftijd;
```

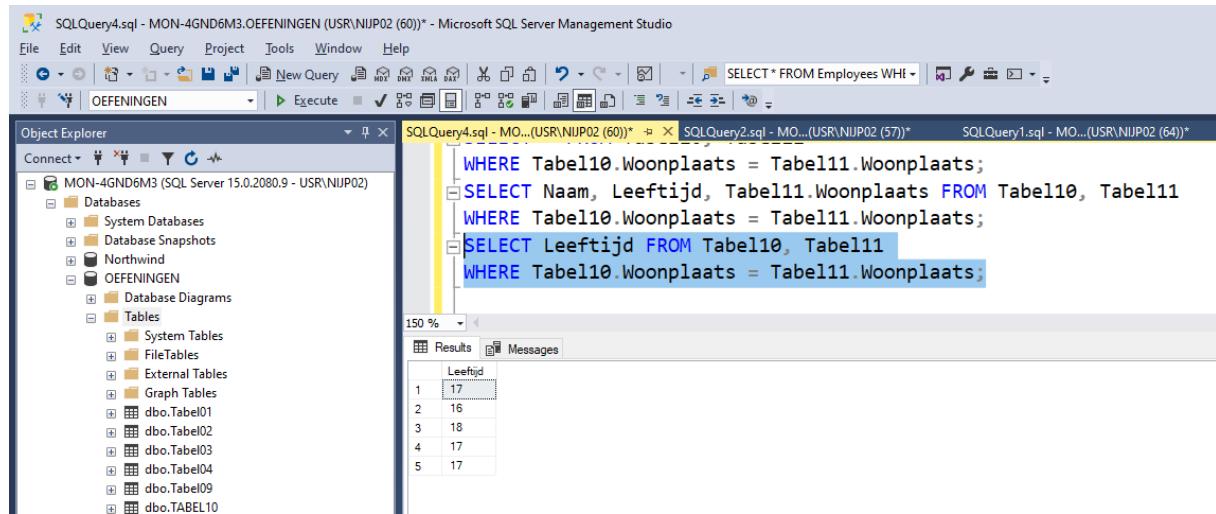


	Naam	Leeftijd	Woonplaats	Naam	Leeftijd	Woonplaats
1	Jan	16	Den Haag	Jan	16	Den Haag

In dit voorbeeld wordt 2x de tabel Tabel11 gebruikt en moet in de query duidelijk zijn welke kolommen uit welke kopie van de tabel moet worden gebruikt. Als een FROM-component twee tabellen met dezelfde naam bevat, moet minimaal één pseudoniem worden gebruikt.

Wanneer we gegevens van twee of meerdere tabellen samenvoegen tot één tabel, dan kan het voorkomen dat het resultaat dubbele waarden bevat. Bij voorbeeld in de query:

```
SELECT Leeftijd FROM Tabel10, Tabel11
WHERE Tabel10.Woonplaats = Tabel11.Woonplaats;
```



The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the database 'OEFENINGEN' is selected. In the center pane, a query window displays the following code:

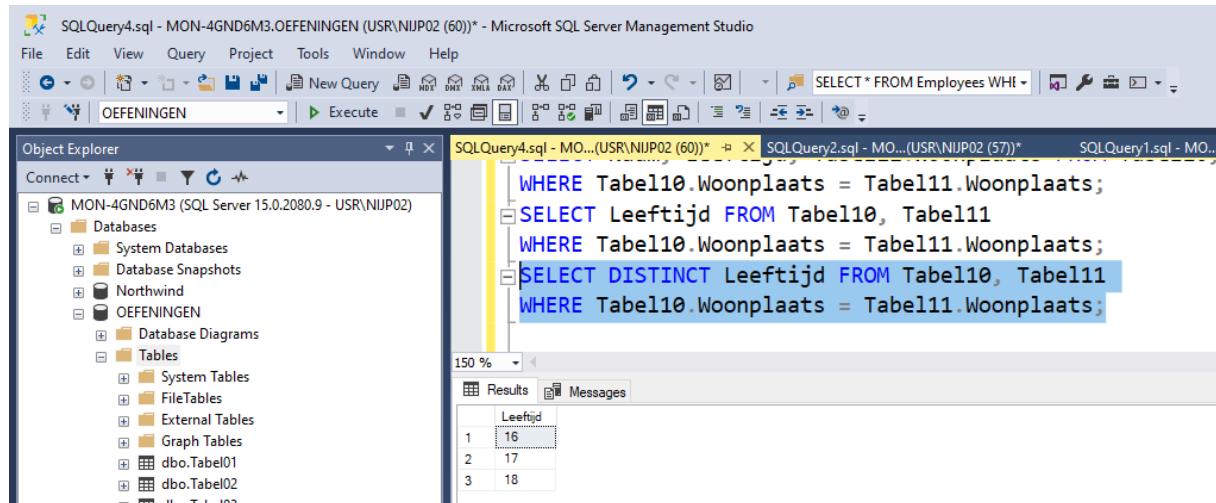
```
WHERE Tabel10.Woonplaats = Tabel11.Woonplaats;
SELECT Naam, Leeftijd, Tabel11.Woonplaats FROM Tabel10, Tabel11
WHERE Tabel10.Woonplaats = Tabel11.Woonplaats;
SELECT Leeftijd FROM Tabel10, Tabel11
WHERE Tabel10.Woonplaats = Tabel11.Woonplaats;
```

The results pane shows the following data:

Leeftijd
17
16
18
17
17

In het resultaat zien we meerdere malen de leeftijd 17 staan. Deze dubbele waarden kunnen we uit het resultaat verwijderen met het woord DISTINCT:

```
SELECT DISTINCT Leeftijd FROM Tabel10, Tabel11
WHERE Tabel10.Woonplaats = Tabel11.Woonplaats;
```



The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the database 'OEFENINGEN' is selected. In the center pane, a query window displays the following code:

```
WHERE Tabel10.Woonplaats = Tabel11.Woonplaats;
SELECT Leeftijd FROM Tabel10, Tabel11
WHERE Tabel10.Woonplaats = Tabel11.Woonplaats;
SELECT DISTINCT Leeftijd FROM Tabel10, Tabel11
WHERE Tabel10.Woonplaats = Tabel11.Woonplaats;
```

The results pane shows the following data:

Leeftijd
16
17
18

De expliciete joins kunnen we onderverdelen in een aantal typen expliciete joins:

- INNER JOIN
- LEFT OUTER JOIN
- RIGHT OUTER JOIN
- FULL OUTER JOIN
- CROSS JOIN

## 15.4 INNER JOIN

Een INNER JOIN geeft alle rijen weer waarvan de waarden in de join kolommen uit beide tabellen gelijk zijn aan elkaar. De waarden uit de eerste tabel die niet voorkomen in de tweede tabel en andersom komen bij een INNER JOIN niet in het resultaat.

De hierboven gebruikte impliciete join:

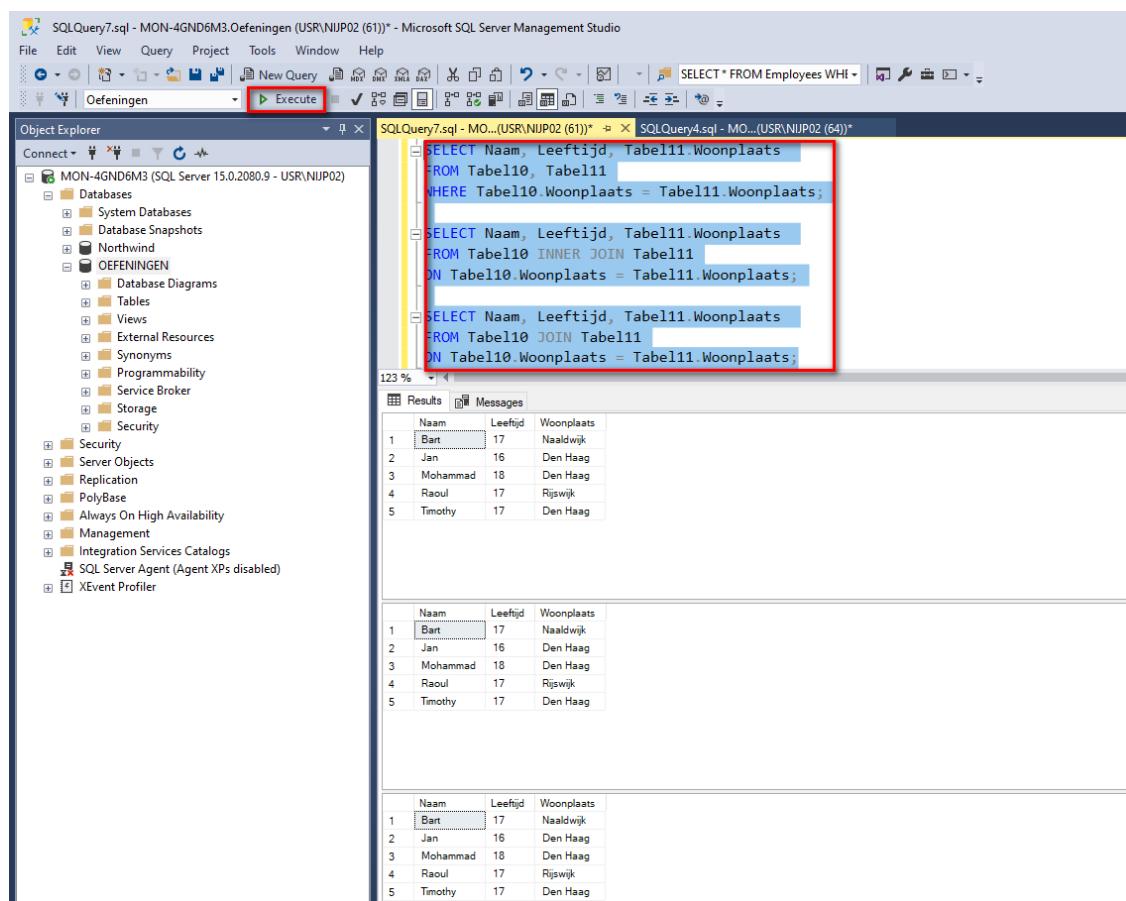
```
SELECT Naam, Leeftijd, Tabel11.Woonplaats
FROM Tabel10, Tabel11
WHERE Tabel10.Woonplaats = Tabel11.Woonplaats;
```

kunnen we ook maken m.b.v. een expliciete inner join met hetzelfde resultaat:

```
SELECT Naam, Leeftijd, Tabel11.Woonplaats
FROM Tabel10 INNER JOIN Tabel11
ON Tabel10.Woonplaats = Tabel11.Woonplaats;
```

Het woord INNER mag ook worden weggelaten. Default (Standaard) wordt er dus een INNER JOIN uitgevoerd wanneer we alleen het woord JOIN in de query zetten. Het woord INNER is alleen toegevoegd om voor de lezer duidelijk aan te geven welke soort JOIN er uitgevoerd zal worden:

```
SELECT Naam, Leeftijd, Tabel11.Woonplaats
FROM Tabel10 JOIN Tabel11
ON Tabel10.Woonplaats = Tabel11.Woonplaats;
```



The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows a database named 'MON-4GND6M3'. The central pane contains three queries, each highlighted with a red box. The first query is an implicit join:

```
SELECT Naam, Leeftijd, Tabel11.Woonplaats
FROM Tabel10, Tabel11
WHERE Tabel10.Woonplaats = Tabel11.Woonplaats;
```

The second query is an explicit INNER JOIN:

```
SELECT Naam, Leeftijd, Tabel11.Woonplaats
FROM Tabel10 INNER JOIN Tabel11
ON Tabel10.Woonplaats = Tabel11.Woonplaats;
```

The third query is another explicit JOIN:

```
SELECT Naam, Leeftijd, Tabel11.Woonplaats
FROM Tabel10 JOIN Tabel11
ON Tabel10.Woonplaats = Tabel11.Woonplaats;
```

The 'Results' pane at the bottom shows the same five rows of data for all three queries, indicating they produce identical results. The data is as follows:

	Naam	Leeftijd	Woonplaats
1	Bart	17	Naaldwijk
2	Jan	16	Den Haag
3	Mohammad	18	Den Haag
4	Raoul	17	Rijswijk
5	Timothy	17	Den Haag

Het verschil tussen de eerste en de tweede query is dat de join conditie (Tabel10.Woonplaats = Tabel11.Woonplaats) verplaatst is van de WHERE-component naar de FROM-component. De verwerking wordt hierdoor ook verplaatst van de WHERE-component naar de FROM-component en wordt de overblijvende WHERE-component eenvoudiger.

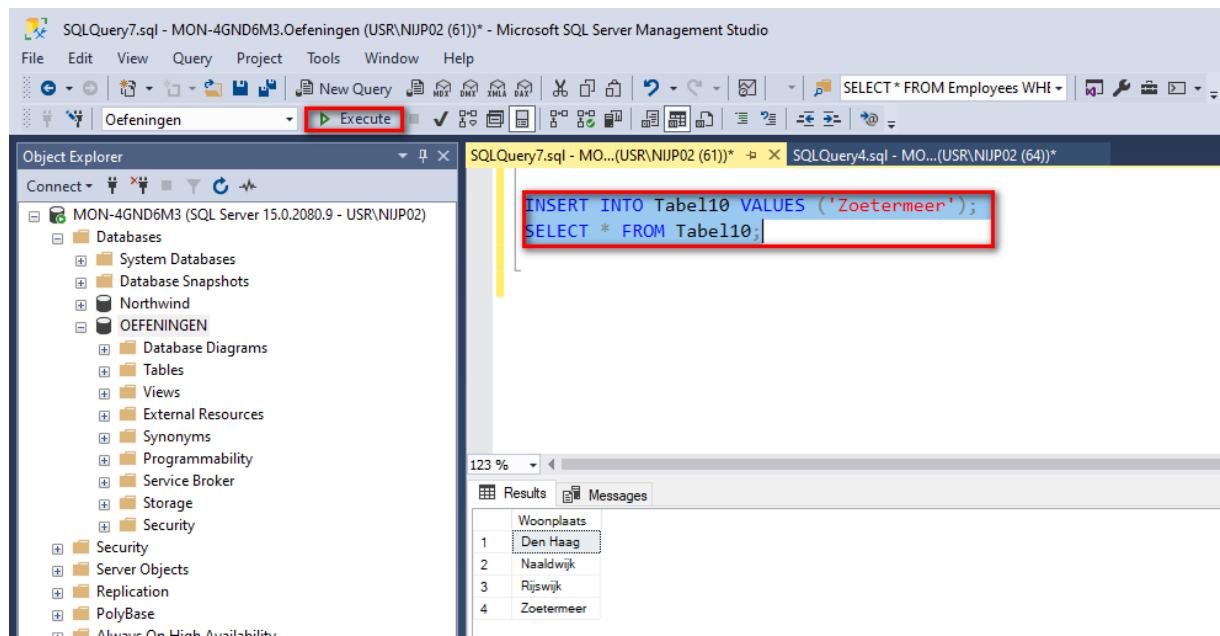
## 15.5 LEFT OUTER JOIN

Een OUTER JOIN kan ook de rijen in het resultaat weergeven waarvan de waarde in een kolom wel in de ene tabel maar niet in de andere tabel voorkomt.

De bovenstaande INNER JOIN geeft als resultaat alle rijen waarvan de woonplaats voorkomt in beide tabellen. Een eventuele woonplaats in Tabel10 waar géén personen uit Tabel11 wonen komt niet in het resultaat terecht.

Stel nu dat we een woonplaats Zoetermeer toevoegen aan Tabel10:

```
INSERT INTO Tabel10 VALUES ('Zoetermeer');
SELECT * FROM Tabel10;
```



The screenshot shows the Microsoft SQL Server Management Studio interface. In the center, there is a query editor window with two statements:

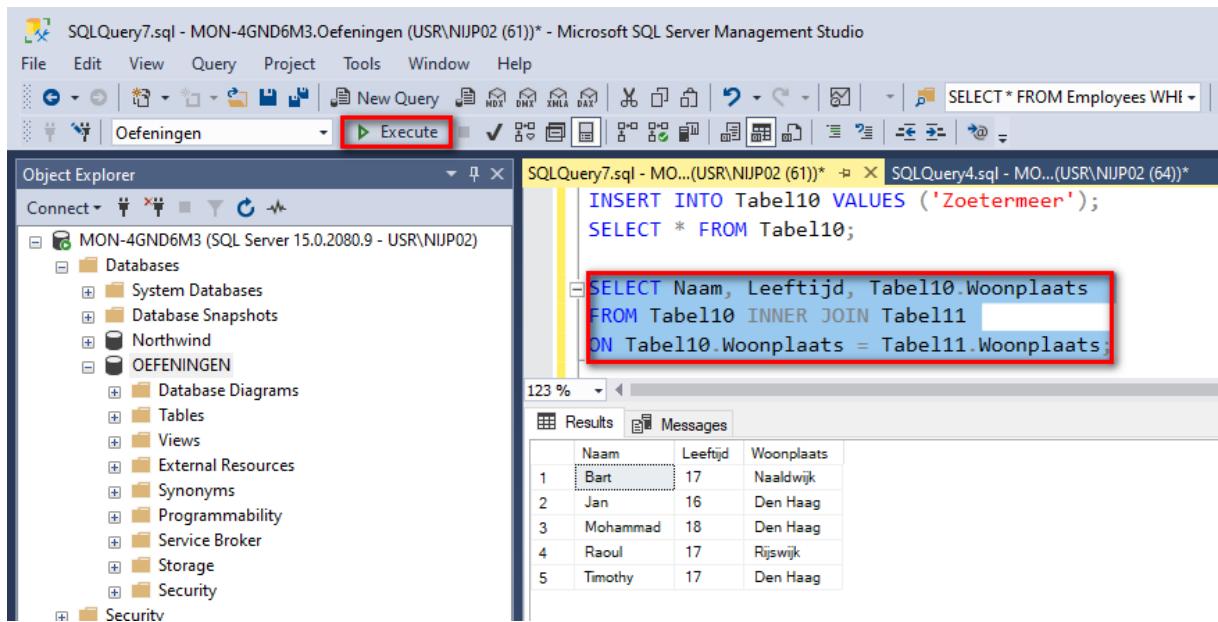
```
INSERT INTO Tabel10 VALUES ('Zoetermeer');
SELECT * FROM Tabel10;
```

The second statement is highlighted with a red box. Below the query editor, the results pane displays a table with one row:

Woonplaats
Den Haag
Naaldwijk
Rijswijk
Zoetermeer

Dan geeft de volgende INNER JOIN het resultaat zonder de woonplaats Zoetermeer:

```
SELECT Naam, Leeftijd, Tabel10.Woonplaats
FROM Tabel10 INNER JOIN Tabel11
ON Tabel10.Woonplaats = Tabel11.Woonplaats;
```



SQLQuery7.sql - MON-4GND6M3.Oefeningen (USR\NIJP02 (61)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Oefeningen Execute

Object Explorer

MON-4GND6M3 (SQL Server 15.0.2080.9 - USR\NIJP02)

- Databases
  - System Databases
  - Database Snapshots
  - Northwind
- OEFENINGEN
  - Database Diagrams
  - Tables
  - Views
  - External Resources
  - Synonyms
  - Programmability
  - Service Broker
  - Storage
  - Security
- Security

SQLQuery7.sql - MO...(USR\NIJP02 (61)) - SQLQuery4.sql - MO...(USR\NIJP02 (64))

```
INSERT INTO Tabel10 VALUES ('Zoetermeer');
SELECT * FROM Tabel10;

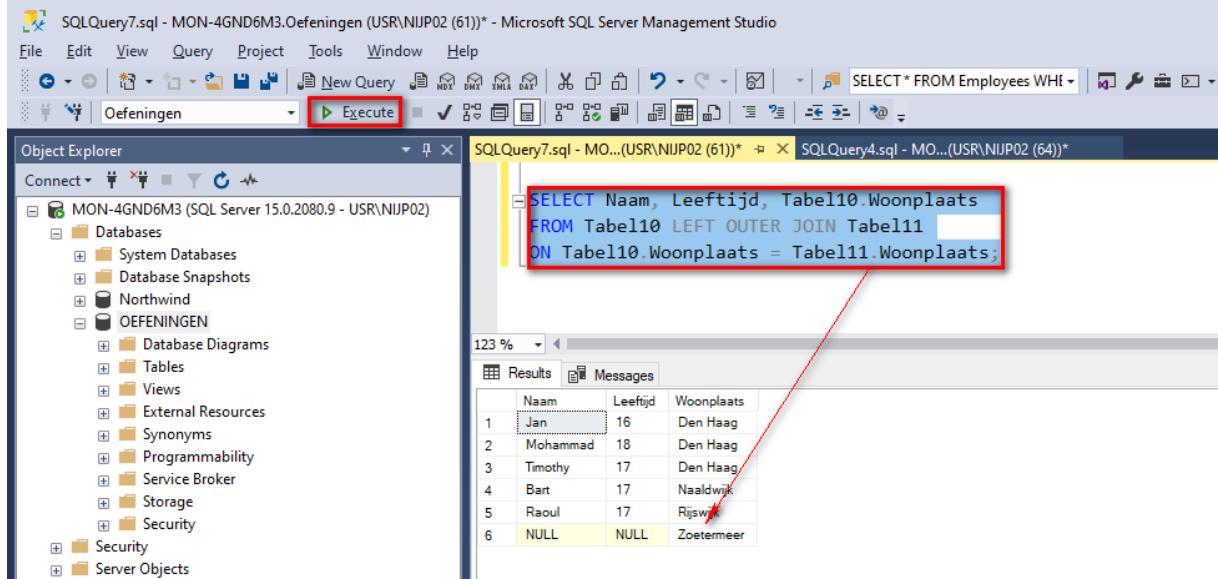
SELECT Naam, Leeftijd, Tabel10.Woonplaats
FROM Tabel10 INNER JOIN Tabel11
ON Tabel10.Woonplaats = Tabel11.Woonplaats;
```

Results

	Naam	Leeftijd	Woonplaats
1	Bart	17	Naaldwijk
2	Jan	16	Den Haag
3	Mohammad	18	Den Haag
4	Raoul	17	Rijswijk
5	Timothy	17	Den Haag

Met een OUTER JOIN kunnen we ook de woonplaatsen uit Tabel10 waar géén van de personen in Tabel11 wonen wél zichtbaar maken in het resultaat.

```
SELECT Naam, Leeftijd, Tabel10.Woonplaats
FROM Tabel10 LEFT OUTER JOIN Tabel11
ON Tabel10.Woonplaats = Tabel11.Woonplaats;
```



SQLQuery7.sql - MON-4GND6M3.Oefeningen (USR\NIJP02 (61)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Oefeningen Execute

Object Explorer

MON-4GND6M3 (SQL Server 15.0.2080.9 - USR\NIJP02)

- Databases
  - System Databases
  - Database Snapshots
  - Northwind
- OEFENINGEN
  - Database Diagrams
  - Tables
  - Views
  - External Resources
  - Synonyms
  - Programmability
  - Service Broker
  - Storage
  - Security
- Server Objects

SQLQuery7.sql - MO...(USR\NIJP02 (61)) - SQLQuery4.sql - MO...(USR\NIJP02 (64))

```
SELECT Naam, Leeftijd, Tabel10.Woonplaats
FROM Tabel10 LEFT OUTER JOIN Tabel11
ON Tabel10.Woonplaats = Tabel11.Woonplaats;
```

Results

	Naam	Leeftijd	Woonplaats
1	Jan	16	Den Haag
2	Mohammad	18	Den Haag
3	Timothy	17	Den Haag
4	Bart	17	Naaldwijk
5	Raoul	17	Rijswijk
6	NULL	NULL	Zoetermeer

Met LEFT OUTER wordt aangegeven dat we in het resultaat ook de rijen willen zien uit de tabel aan de linkerzijde van het woord JOIN waarvan de waarde in de kolom niet voorkomt in kolom van de tabel aan de rechterzijde van het woord JOIN. In bovenstaand voorbeeld willen we dus wel de waarden van de Woonplaats uit Tabel10 zien waarvan de woonplaats niet voorkomt in de kolom Woonplaats van Tabel11. Als resultaat zien we dan dus ook de Woonplaats Zoetermeer met NULL waarden in de kolommen Naam en Leeftijd omdat Zoetermeer niet voorkomt in Tabel11.

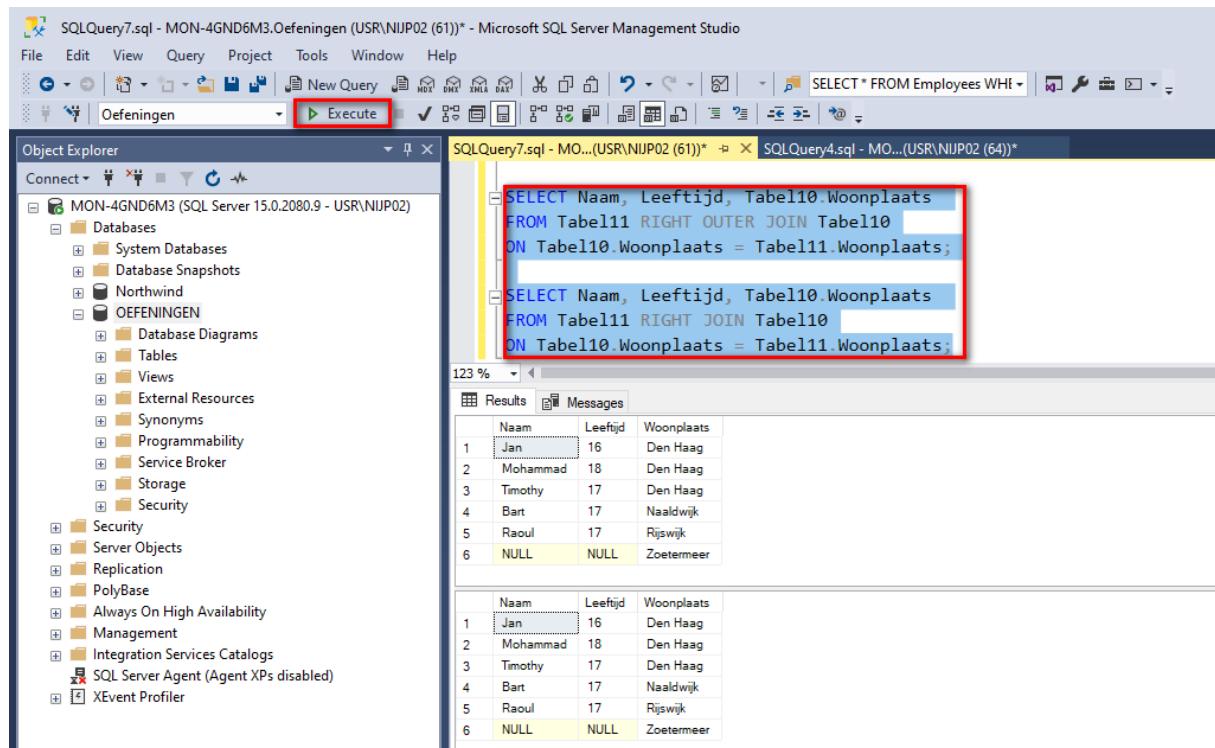
## 15.6 RIGHT OUTER JOIN

Op eenzelfde manier kunnen we met een RIGHT OUTER JOIN ook de waarden in de tabel aan de rechterzijde van het woord JOIN weergeven die niet in de tabel aan de linkerzijde van het woord JOIN voorkomen.

```
SELECT Naam, Leeftijd, Tabel10.Woonplaats
FROM Tabel11 RIGHT OUTER JOIN Tabel10
ON Tabel10.Woonplaats = Tabel11.Woonplaats;
```

Het woord OUTER mogen we eventueel ook weglaten omdat het woord LEFT of RIGHT ook reeds aangeven dat het een OUTER JOIN betreft:

```
SELECT Naam, Leeftijd, Tabel10.Woonplaats
FROM Tabel11 RIGHT JOIN Tabel10
ON Tabel10.Woonplaats = Tabel11.Woonplaats;
```



	Naam	Leeftijd	Woonplaats
1	Jan	16	Den Haag
2	Mohammad	18	Den Haag
3	Timothy	17	Den Haag
4	Bart	17	Naaldwijk
5	Raoul	17	Rijswijk
6	NULL	NULL	Zoetermeer

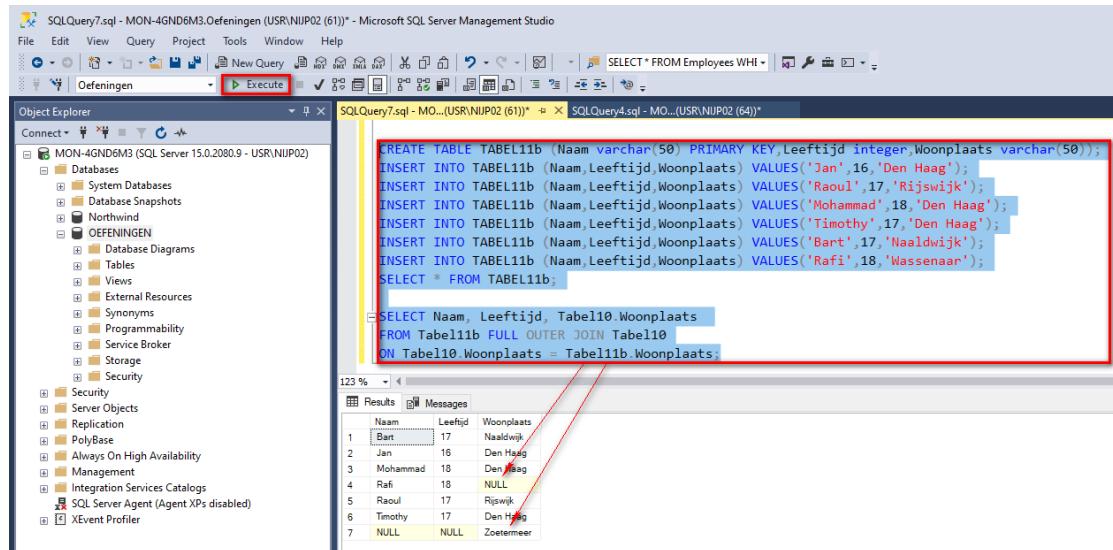
## 15.7 FULL OUTER JOIN

De FULL OUTER JOIN is een combinatie van een LEFT OUTER JOIN en een RIGHT OUTER JOIN. Met een FULL OUTER JOIN kunnen we zowel de waarden in de kolom van de tabel aan de linkerzijde van het woord JOIN alsmede de waarden in de kolom aan de rechterzijde van het woord JOIN, die niet aan de andere zijde van het woord JOIN aanwezig zijn zichtbaar maken in het resultaat:

```
CREATE TABLE TABEL11b (Naam varchar(50) PRIMARY KEY, Leeftijd integer, Woonplaats
varchar(50));
INSERT INTO TABEL11b (Naam, Leeftijd, Woonplaats) VALUES('Jan', 16, 'Den Haag');
INSERT INTO TABEL11b (Naam, Leeftijd, Woonplaats) VALUES('Raoul', 17, 'Rijswijk');
INSERT INTO TABEL11b (Naam, Leeftijd, Woonplaats) VALUES('Mohammad', 18, 'Den Haag');
INSERT INTO TABEL11b (Naam, Leeftijd, Woonplaats) VALUES('Timothy', 17, 'Den Haag');
INSERT INTO TABEL11b (Naam, Leeftijd, Woonplaats) VALUES('Bart', 17, 'Naaldwijk');
```

```
INSERT INTO TABEL11b (Naam, Leeftijd, Woonplaats) VALUES('Rafi', 18, 'Wassenaar');
SELECT * FROM TABEL11b;
```

```
SELECT Naam, Leeftijd, Tabel10.Woonplaats
FROM Tabel11b FULL OUTER JOIN Tabel10
ON Tabel10.Woonplaats = Tabel11b.Woonplaats;
```



The screenshot shows the SQL Server Management Studio interface. The Object Explorer on the left shows the database structure. The main window contains a query editor with the following code:

```
CREATE TABLE TABEL11b (Naam varchar(50) PRIMARY KEY, Leeftijd integer, Woonplaats varchar(50));
INSERT INTO TABEL11b (Naam, Leeftijd, Woonplaats) VALUES('Jan', 16, 'Den Haag');
INSERT INTO TABEL11b (Naam, Leeftijd, Woonplaats) VALUES('Raoul', 17, 'Rijswijk');
INSERT INTO TABEL11b (Naam, Leeftijd, Woonplaats) VALUES('Mohammad', 18, 'Den Haag');
INSERT INTO TABEL11b (Naam, Leeftijd, Woonplaats) VALUES('Timothy', 17, 'Den Haag');
INSERT INTO TABEL11b (Naam, Leeftijd, Woonplaats) VALUES('Bart', 17, 'Naaldwijk');
INSERT INTO TABEL11b (Naam, Leeftijd, Woonplaats) VALUES('Rafi', 18, 'Wassenaar');

SELECT Naam, Leeftijd, Tabel10.Woonplaats
FROM Tabel11b FULL OUTER JOIN Tabel10
ON Tabel10.Woonplaats = Tabel11b.Woonplaats;
```

The results grid displays the following data:

	Naam	Leeftijd	Woonplaats
1	Bart	17	Naaldwijk
2	Jan	16	Den Haag
3	Mohammad	18	Den Haag
4	Rafi	18	NULL
5	Raoul	17	Rijswijk
6	Timothy	17	Den Haag
7	NULL	NULL	Zoetermeer

De persoon Rafi in Tabel11b woont in Wassenaar, maar Wassenaar komt niet voor in tabel10. Woonplaats Zoetermeer in Tabel10 komt niet voor in Tabel11. Aan beide zijden van de join clausule staan dus waarden die in de andere tabel niet voorkomt. Door de FULL OUTER JOIN worden beiden zichtbaar. De FULL OUTER JOIN is dus een combinatie van een LEFT OUTER JOIN en een RIGHT OUTER JOIN.

## 15.8 CROSS JOIN

Een CROSS JOIN heeft een Cartesisch Product van beide tabellen als resultaat.

```
SELECT *
FROM Tabel10 CROSS JOIN Tabel11;
```

SQLQuery7.sql - MON-4GND6M3.Oefeningen (USR\NJP02 (61)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Execute (button highlighted with red box)

Object Explorer

MON-4GND6M3 (SQL Server 15.0.2080.9 - USR\NJP02)

- Databases
  - System Databases
  - Database Snapshots
  - Northwind
  - OEFENINGEN
  - Tables
  - Views
  - External Resources
  - Synonyms
  - Programmability
  - Service Broker
  - Storage
  - Security
- Security
- Server Objects
- Replication
- PolyBase
- Always On High Availability
- Management
- Integration Services Catalogs
- SQL Server Agent (Agent XPs disabled)
- XEvent Profiler

SQLQuery7.sql - MO... (USR\NJP02 (61))

SQLQuery4.sql - MO... (USR\NJP02 (64))

```
SELECT *  
FROM Tabel10 CROSS JOIN Tabel11;
```

Results

	Woonplaats	Naam	Leeftijd	Woonplaats
1	Den Haag	Bart	17	Naaldwijk
2	Den Haag	Jan	16	Den Haag
3	Den Haag	Mohammad	18	Den Haag
4	Den Haag	Raoul	17	Rijswijk
5	Den Haag	Timothy	17	Den Haag
6	Naaldwijk	Bart	17	Naaldwijk
7	Naaldwijk	Jan	16	Den Haag
8	Naaldwijk	Mohammad	18	Den Haag
9	Naaldwijk	Raoul	17	Rijswijk
10	Naaldwijk	Timothy	17	Den Haag
11	Rijswijk	Bart	17	Naaldwijk
12	Rijswijk	Jan	16	Den Haag
13	Rijswijk	Mohammad	18	Den Haag
14	Rijswijk	Raoul	17	Rijswijk
15	Rijswijk	Timothy	17	Den Haag
16	Zoetermeer	Bart	17	Naaldwijk
17	Zoetermeer	Jan	16	Den Haag
18	Zoetermeer	Mohammad	18	Den Haag
19	Zoetermeer	Raoul	17	Rijswijk
20	Zoetermeer	Timothy	17	Den Haag

## Hoofdstuk 16 SEQUENCES – (NEW)

### 16.1 Sequences

Veel tabellen die gecreëerd worden hebben een kolom met unieke nummers. Deze nummers worden gebruikt om de rijen te identificeren en vormen meestal de primaire sleutel van de tabel. Elk nummer in de kolom is dan uniek. Telkens als we een rij willen toevoegen dient er een nummer gegenereerd te worden. Met sequences kunnen we nummers genereren.

Een sequence is een onafhankelijk object in de database. Telkens wanneer je een nieuwe waarde moet genereren dan roep je een functie aan op de sequence en gebruik je de door de functie teruggegeven waarde.

Een sequence kan worden gemaakt met de syntax:

```
CREATE SEQUENCE <sequencenaam>
[AS <datatype>]
[START WITH <constante>]
[INCREMENT BY <constante>]
[MINVALUE <constante> | NOMINVALUE]
[MAXVALUE <constante> | NOMAXVALUE]
[CYCLE | NOCYCLE]
[CACHE <constante> | NOCACHE]
```

We kunnen een nieuw nummer genereren met:

```
SELECT NEXT VALUE FOR <sequencenaam>;
```

We kunnen de waarde uit de sequence gebruiken bij het invoeren van een rij:

```
INSERT INTO <tabelnaam> VALUES (SELECT NEXT VALUE FOR <sequencenaam>, ...);
```

Voorbeeld:

Een sequence kan worden verwijderd met de syntax:

```
DROP SEQUENCE <sequencenaam>
```

Voorbeeld:

We kunnen bijvoorbeeld een sequence maken die de OrderID's genereert.

```
USE Northwind;
CREATE SEQUENCE dbo.SeqOrderIDs AS INT
MINVALUE 1
Cycle;

SELECT NEXT VALUE FOR dbo.SeqOrderIDs;
SELECT NEXT VALUE FOR dbo.SeqOrderIDs;
SELECT NEXT VALUE FOR dbo.SeqOrderIDs;
```

SQLQuery11.sql - M...(USR\NIJP02 (63))\* × SQLQuery10.sql - M...(USR\NIJP02 (57))\*

```

USE Northwind;
CREATE SEQUENCE dbo.SeqOrderIDs AS INT
    MINVALUE 1
    Cycle;

SELECT NEXT VALUE FOR dbo.SeqOrderIDs;
SELECT NEXT VALUE FOR dbo.SeqOrderIDs;
SELECT NEXT VALUE FOR dbo.SeqOrderIDs;

```

179 %

	Results	Messages
1	(No column name)	
1	1	
1	(No column name)	
1	2	
1	(No column name)	
1	3	

Telkens wanneer de functie NEXT VALUE wordt uitgevoerd dan wordt er een nieuw nummer gegenereerd. Het eerste nummer dat er gegenereerd wordt is nummer 1, daarna nummer 2 etc.

We kunnen de waarde van de laatst uitgegeven nummer van de sequence opvragen met:

```

SELECT current_value
FROM sys.sequences
WHERE OBJECT_ID = OBJECT_ID(N'dbo.SeqOrderIDs');

```

SQLQuery11.sql - M...(USR\NIJP02 (63))\* × SQLQuery10.sql - M...(USR\NIJP02 (57))\*

```

USE Northwind;
CREATE SEQUENCE dbo.SeqOrderIDs AS INT
    MINVALUE 1
    Cycle;

SELECT NEXT VALUE FOR dbo.SeqOrderIDs;
SELECT NEXT VALUE FOR dbo.SeqOrderIDs;
SELECT NEXT VALUE FOR dbo.SeqOrderIDs;

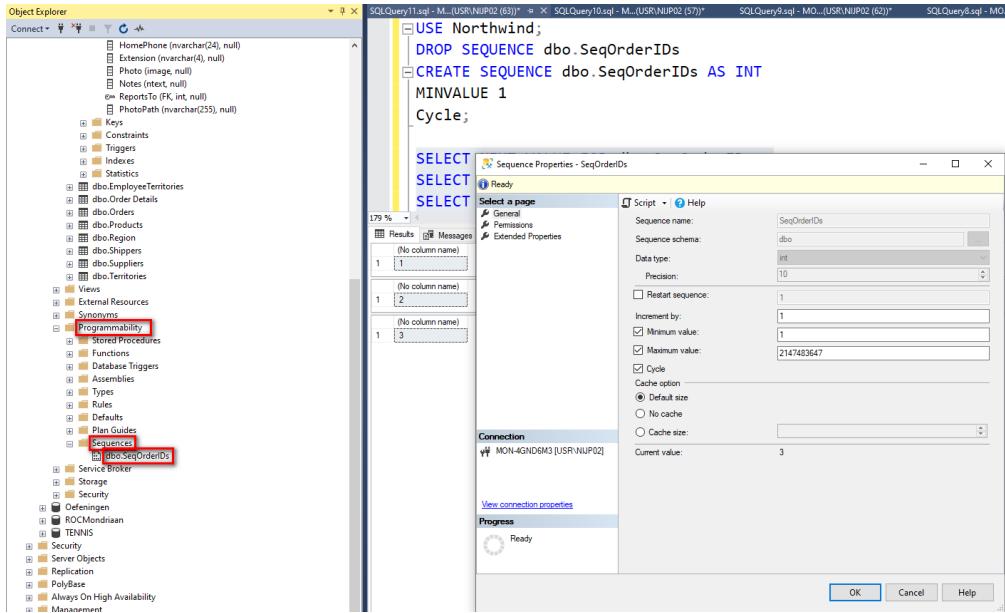
SELECT current_value
FROM sys.sequences
WHERE OBJECT_ID = OBJECT_ID(N'dbo.SeqOrderIDs');

```

179 %

	Results	Messages
1	current_value	
1	3	

De gecreëerde sequence is in SSMS terug te vinden in de boom:



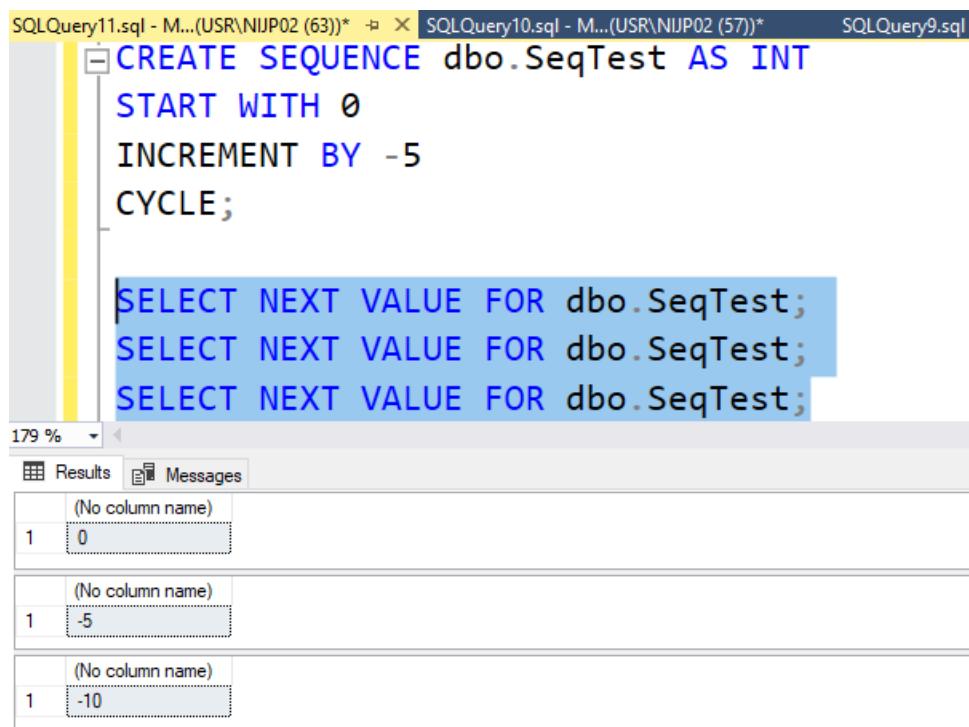
The screenshot shows the SSMS interface. In the Object Explorer on the left, under the 'Programmability' node, there is a 'Sequences' folder which contains a sequence named 'SeqOrderIDs'. A red box highlights this sequence in the tree view. To the right, the 'Sequence Properties - SeqOrderIDs' dialog box is open. It shows the following settings:

- Sequence name:** SeqOrderIDs
- Sequence schema:** dbo
- Data type:** INT
- Precision:** 10
- Increment by:** 1
- Minimum value:** 1
- Maximum value:** 2147483647
- Cycle:** checked
- Cache option:** Default size (radio button selected)
- Current value:** 3

We kunnen nu een sequence maken die grotere stappen maakt in bijvoorbeeld tegengestelde richting:

```
CREATE SEQUENCE dbo.SeqTest AS INT
START WITH 0
INCREMENT BY -5
CYCLE;

SELECT NEXT VALUE FOR dbo.SeqTest;
SELECT NEXT VALUE FOR dbo.SeqTest;
SELECT NEXT VALUE FOR dbo.SeqTest;
```



The screenshot shows the SSMS interface with three tabs at the top: 'SQLQuery11.sql - M... (USR\NIJP02 (63))', 'SQLQuery10.sql - M... (USR\NIJP02 (57))', and 'SQLQuery9.sql'. The code in the first tab is:

```
CREATE SEQUENCE dbo.SeqTest AS INT
START WITH 0
INCREMENT BY -5
CYCLE;
```

The code in the second tab is:

```
SELECT NEXT VALUE FOR dbo.SeqTest;
SELECT NEXT VALUE FOR dbo.SeqTest;
SELECT NEXT VALUE FOR dbo.SeqTest;
```

The results pane shows the output of the third query:

	Results	Messages
1	(No column name) 0	
1	(No column name) -5	
1	(No column name) -10	

## Hoofdstuk 17 PROGRAMMEERBARE OBJECTEN – (NEW)

Binnen SQL is het mogelijk om programmeerbare objecten te gebruiken. In de komende hoofdstukken behandelen we de mogelijkheden in T-SQL, het Microsoft SQL Server dialect van SQL. Net zoals in de voorgaande hoofdstukken zal de SQL gereedschappen m.b.v. voorbeelden worden behandeld. Doel van de hoofdstukken is wederom het leren kennen van de SQL gereedschappen en het gebruik ervan. Omdat het geen programmeerstudie betreft zal de uiteindelijke vaardigheid in het gebruik van de gereedschappen af hangen van de programmeervaardigheden van de student zelf.

### 17.1 Variabelen

Met het DECLARE statement kunnen we één of meerdere variabelen declareren en met SET kunnen we een waarde aan de variabele toekennen.

Voorbeeld:

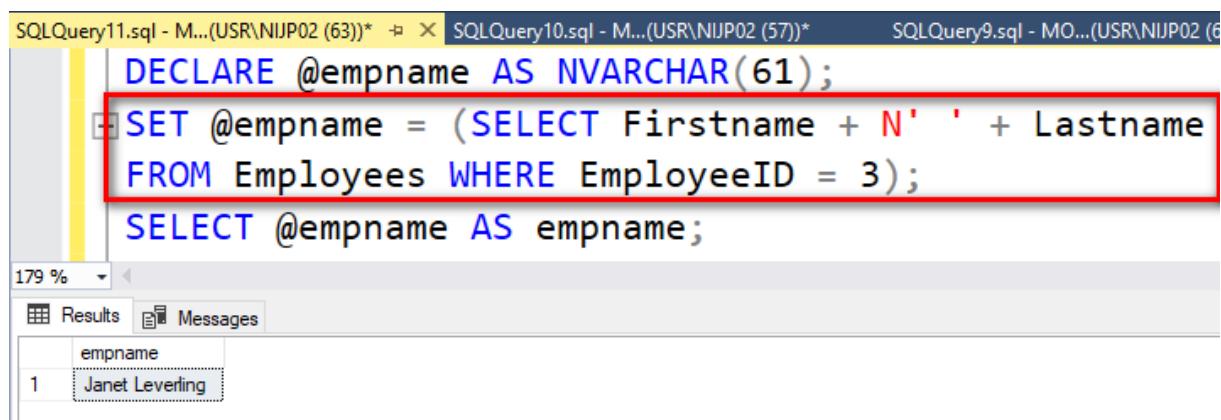
```
DECLARE @i AS INT;
SET @i = 10;
```

Dit mag ook op de volgende alternatieve methode in één regel:

```
DECLARE @i AS INT = 10;
```

We kunnen ook het resultaat van een scalaire expressie aan een variabele toekennen:

```
DECLARE @empname AS NVARCHAR(61);
SET @empname = (SELECT Firstname + N' ' + Lastname FROM Employees WHERE EmployeeID = 3);
SELECT @empname AS empname;
```



empname
Janet Leverling

### 17.2 Batches

Een batch is één of meerdere SQL statements die naar de databaseserver (RDBMS) gestuurd wordt om als één enkele unit uitgevoerd te laten worden. De batch ondergaat **parsing** (controle van de syntax), **resolution/binding** (controle op het bestaan van de gerefereerde objecten en kolommen, controle van de permissies) en **optimalisatie** als een unit.

Een batch is niet hetzelfde als een transactie. Een transactie is een atomaire (ondeelbare) eenheid van arbeid. Een batch kan meerdere transacties bevatten, en een transactie kan verdeeld over meerdere batches worden ingediend. Wanneer een transactie wordt gecanceld of teruggedraaid dan

maakt SQL Server alle deelactiviteiten ongedaan die sinds het begin van de transactie hebben plaats gevonden, ongeacht waar de batch begon.

Client applicaties van SQL Server, zoals SSMS, SSQLCMD, OSQL, voorzien in een client tool commando **GO** die het einde van de batch signaleert. Het GO commando is een client commando en niet en T-SQL commando.

### 17.3 Een batch als een unit of parsing

Een batch is een set commando's die als een unit geparst en uitgevoerd worden. Als de parsing succesvol is, dan zal SQL Server proberen om de batch uit te voeren. Wanneer er een syntax fout in de batch zit, dan wordt de gehele batch niet ingediend aan SQL Server om uit te voeren.

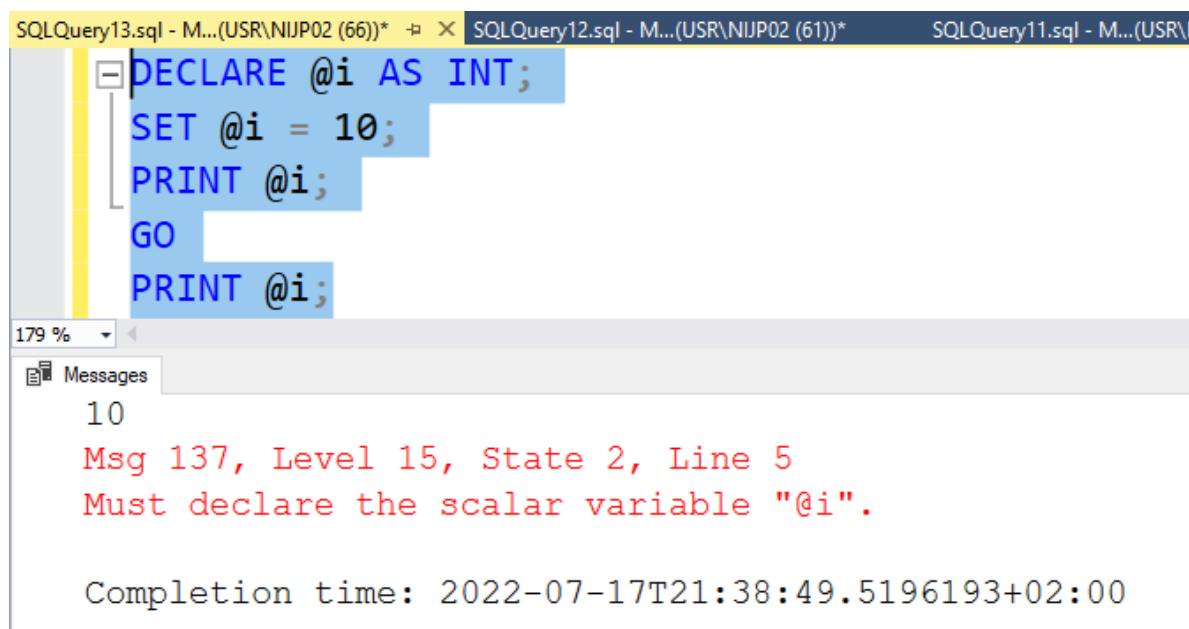
Wanneer bijvoorbeeld drie batches achter elkaar worden uitgevoerd en er in de tweede batch een syntactische fout zit, dan worden de eerste en de derde batch wel uitgevoerd maar de tweede niet en ontstaat er op de plaats van de tweede batch een foutmelding.

### 17.4 Batches en variabelen

Een variabele is lokaal binnen de batch waarin het gedefinieerd is. Wanneer je refereert naar een variabele in een andere batch (bijvoorbeeld de bovenstaande batch) dan krijg je een foutmelding die zegt dat de variabele niet gedefinieerd is.

Voorbeeld:

```
DECLARE @i AS INT;
SET @i = 10;
PRINT @i;
GO
PRINT @i;
```



The screenshot shows a SQL Server Management Studio interface with three tabs at the top: 'SQLQuery13.sql - M...', 'SQLQuery12.sql - M...', and 'SQLQuery11.sql - M...'. The 'SQLQuery13.sql' tab contains the following script:

```
DECLARE @i AS INT;
SET @i = 10;
PRINT @i;
GO
PRINT @i;
```

The first four lines are highlighted in blue, indicating they have been parsed. The fifth line, 'PRINT @i;', is highlighted in red, indicating it has not been parsed due to a syntax error. In the 'Messages' pane below, the output is:

```
10
Msg 137, Level 15, State 2, Line 5
Must declare the scalar variable "@i".
```

Below the messages, the completion time is shown as: Completion time: 2022-07-17T21:38:49.5196193+02:00

De tweede PRINT @i; geeft een foutmelding daar @i gedeclareerd was in de voorgaande batch.

## 17.5 Statements die niet in een batch gecombineerd kunnen worden

De volgende statements kunnen niet met andere statements binnen eenzelfde batch worden gecombineerd: CREATE DEFAULT, CREATE FUNCTION, CREATE PROCEDURE, CREATE RULE, CREATE SCHEMA, CREATE TRIGGER en CREATE VIEW.

Voorbeeld:

```
DROP VIEW IF EXISTS MyView;
CREATE VIEW MyView
AS
SELECT YEAR(orderdate) AS orderyear, COUNT(*) AS numorders
FROM Orders
GROUP BY YEAR(orderdate);
GO
```

**SQLQuery12.sql - M... (USR\NIJP02 (61))\***   **SQLQuery11.sql - M... (USR\NIJP02 (63))\***   **SQLQuery10.sql - M... (USR\NIJP02 (57))\***   **SQLQue**

**DROP VIEW IF EXISTS MyView;**

**CREATE VIEW MyView** AS

**AS**

**SELECT YEAR(orderdate) AS orderyear, COUNT(\*) AS numorders**

**FROM Orders**

**GROUP BY YEAR(orderdate);**

**GO**

179 %

Messages

Msg 111, Level 15, State 1, Line 2  
 'CREATE VIEW' must be the first statement in a query batch.

Completion time: 2022-07-17T21:33:52.3655619+02:00

Doordat de DROP VIEW en CREATE VIEW binnen dezelfde batch vallen ontstaat er een foutmelding:

```
Msg 111, Level 15, State 1, Line 2
'CREATE VIEW' must be the first statement in a query batch.
```

Dit probleem kunnen we voorkomen door de DROP VIEW en de CREATE VIEW van elkaar te scheiden door een GO commando achter de DROP VIEW te zetten (de batch verdelen in twee batches).

## 17.6 Batch als oplossingsunit

Een batch is een oplossingseenheid (binding), dat betekent dat het controleren van het bestaan van de objecten en kolommen op batch niveau gebeurt. Houd hier rekening mee wanneer je batches ontwerpt!

Wanneer je veranderingen aanbrengt aan een object en binnen dezelfde batch de data in het object wilt manipuleren, dan kan het zijn dat SQL Server nog niet op de hoogte is van de schema verandering en daardoor fout loopt.

Voorbeeld:

```

DROP TABLE IF EXISTS T1;
CREATE TABLE T1(col1 INT);
GO
ALTER TABLE T1 ADD col2 INT;
SELECT col1, col2 FROM T1;
SQLQuery14.sql - M...(USR\NIJP02 (74))*  ↳ SQLQuery13.sql - M...(USR\NIJP02 (66))*
SQLQuery12.sql - M...(U

```

DROP TABLE IF EXISTS T1;

CREATE TABLE T1(col1 INT);

GO

ALTER TABLE T1 ADD col2 INT;

SELECT col1, col2 FROM T1;

179 %

Messages

Msg 207, Level 16, State 1, Line 5  
Invalid column name 'col2'.

Completion time: 2022-07-17T22:22:30.1578125+02:00

De beste praktijk om dit te voorkomen is door DDL en DML statements in aparte batches te plaatsen, dus:

```

ALTER TABLE T1 ADD col2 INT;
GO
SELECT col1, col2 FROM T1;
SQLQuery14.sql - M...(USR\NIJP02 (74))*  ↳ SQLQuery13.sql - M...(USR\NIJP02 (66))*
SQLQuery1

```

DROP TABLE IF EXISTS T1;

CREATE TABLE T1(col1 INT);

GO

ALTER TABLE T1 ADD col2 INT;

GO

SELECT col1, col2 FROM T1;

179 %

Results Messages

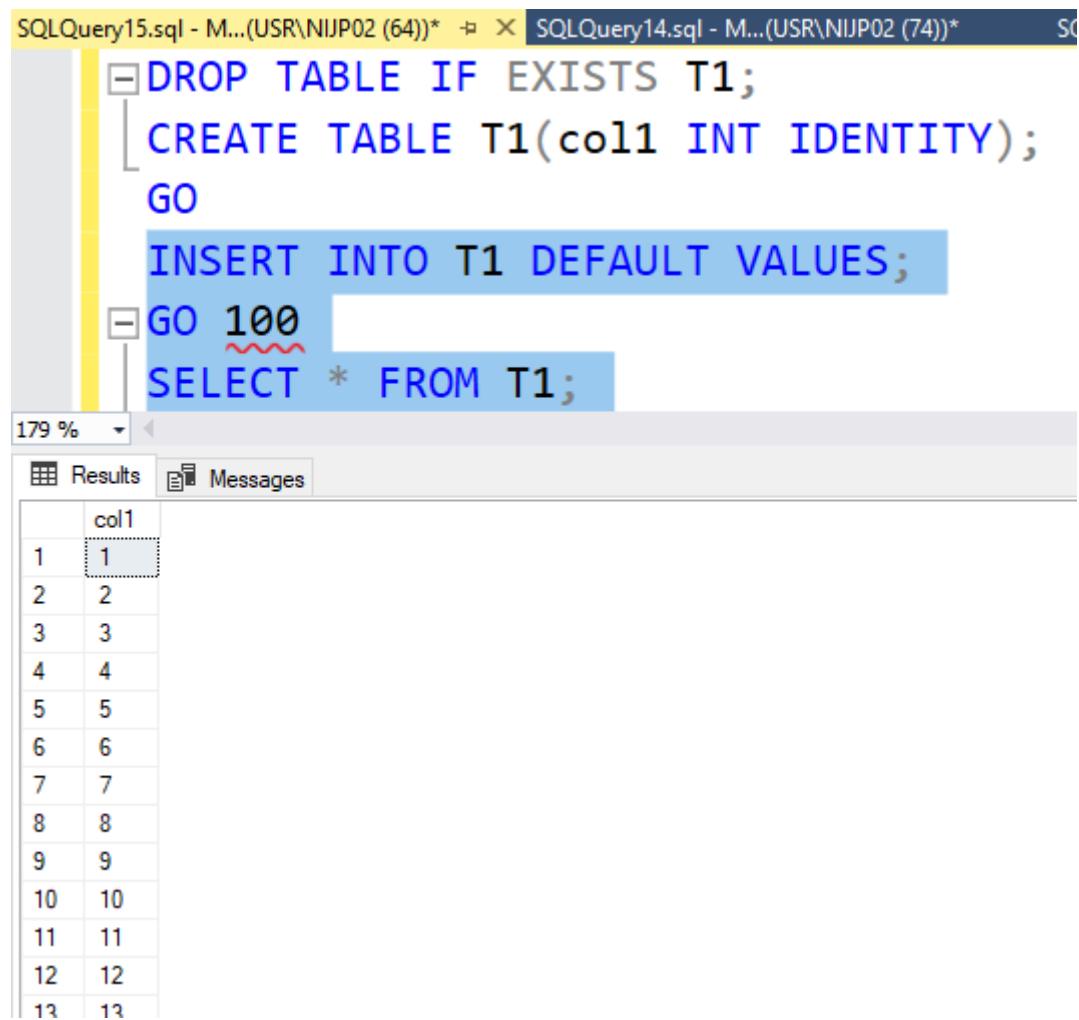
col1	col2
------	------

## 17.7 De GO n optie

GO is geen T-SQL commando maar een commando dat gebruikt wordt door SQL Servers Client tools, zoals SSMS om het einde van een batch aan te geven. Dit commando ondersteunt een argument die aangeeft hoe vaak de batch uitgevoerd moet worden.

Voorbeeld:

```
DROP TABLE IF EXISTS T1;
CREATE TABLE T1(col1 INT IDENTITY);
GO
INSERT INTO T1 DEFAULT VALUES;
GO 100
SELECT * FROM T1;
```



	col1
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13

## 17.8 IF ... ELSE

Met het IF ... ELSE element kan de stroom van de code worden geregeld op basis van het resultaat van een predikaat (uitdrukking). Er wordt een code blok gespecificeerd die uitgevoerd moet worden als het predikaat TRUE is en er wordt een code block gespecificeerd die uitgevoerd moet worden als het predikaat FALSE of UNKNOWN is.

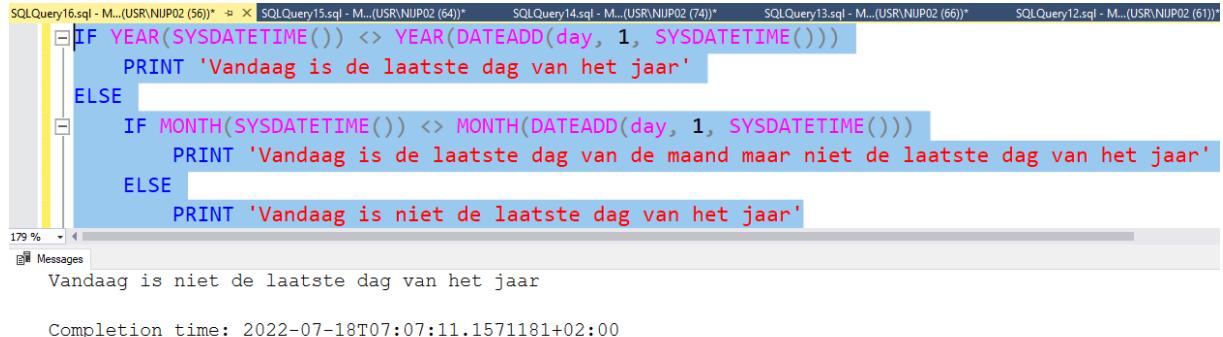
Let er op dat SQL een drie-waarden logica gebruikt en het ELSE blok uitgevoerd wordt wanneer het predikaat fout of onbekend is. Maak daarom een expliciete test voor NULL waarden met een IS NULL predikaat.

Voorbeeld:

```

IF YEAR(SYSDATETIME()) <> YEAR(DATEADD(day, 1, SYSDATETIME()))
    PRINT 'Vandaag is de laatste dag van het jaar'
ELSE
    IF MONTH(SYSDATETIME()) <> MONTH(DATEADD(day, 1, SYSDATETIME()))
        PRINT 'Vandaag is de laatste dag van de maand maar niet de laatste dag
van het jaar'
    ELSE
        PRINT 'Vandaag is niet de laatste dag van het jaar'

```



The screenshot shows a SQL Server Management Studio window with multiple tabs at the top. The main area contains the T-SQL code. The output pane at the bottom shows the result of the execution: "Vandaag is niet de laatste dag van het jaar". Below the output, the completion time is displayed as "Completion time: 2022-07-18T07:07:11.1571181+02:00".

Wanneer er meerdere statements in de IF of ELSE uitgevoerd moet worden dan moet een statement blok worden gebruikt die begin met **BEGIN** en eindigt met **END**.

## 17.9 WHILE

T-SQL heeft een **WHILE** element om in de code een **loop** (lus) op te nemen. Het WHILE element voert een statement of statementblok herhaald uit, zolang het predicaat die je achter het WHILE keyword specificeert TRUE is. Wanneer het predicaat FALSE of UNKNOWN is wordt de loop afgesloten.

T-SQL heeft géén build-in loop element die een berekend aantal keren uitgevoerd wordt, maar dat is met een WHILE na te bootsen.

Voorbeeld:

Onderstaand voorbeeld wordt 10 maal uitgevoerd:

```

DECLARE @i AS INT = 1;
WHILE @i <= 10
BEGIN
    PRINT @i;
    SET @i = @i + 1;
END;

```

SQLQuery16.sql - M...(USR\NIJP02 (56))\*

```

DECLARE @i AS INT = 1;
WHILE @i <= 10
BEGIN
    PRINT @i;
    SET @i = @i + 1;
END;

```

179 % Messages

```

1
2
3
4
5
6
7
8
9
10

```

Completion time: 2022-07-18T09:32:00.0720173+02:00

Wanneer je op een punt in een loop de loop wilt breken en verder wil gaan met het statement die op de loop volgt dan kan gebruik worden gemaakt van een **BREAK** commando.

Voorbeeld:

```

DECLARE @i AS INT = 1;
WHILE @i <= 10
BEGIN
    IF @i = 6 BREAK;
    PRINT @i;
    SET @i = @i + 1;
END;

```

SQLQuery16.sql - M...(USR\NIJP02 (56))\*

```

DECLARE @i AS INT = 1;
WHILE @i <= 10
BEGIN
    IF @i = 6 BREAK;
    PRINT @i;
    SET @i = @i + 1;
END;

```

179 %

Messages

```

1
2
3
4
5

```

Completion time: 2022-07-18T09:37:31.6721444+02:00

Wanneer je op een bepaald punt in de body van de loop de rest van de activiteiten wilt overslaan en het loop predicaat wilt evalueren kan **CONTINUE** worden gebruikt.

Voorbeeld:

In het onderstaand voorbeeld wordt de activiteit van de 6<sup>e</sup> iteratie van de loop vanaf het punt van de IF overgeslagen tot het einde van het loopblok.

```

DECLARE @i AS INT = 0;
WHILE @i < 10
BEGIN
    SET @i = @i + 1;
    IF @i = 6 CONTINUE;
    PRINT @i;
END;

```

SQLQuery16.sql - M...(USR\NIJP02 (56))\*

```

DECLARE @i AS INT = 0;
WHILE @i < 10
BEGIN
    SET @i = @i + 1;
    IF @i = 6 CONTINUE;
    PRINT @i;
END;

```

179 %

Messages

```

1
2
3
4
5
6
7
8
9
10

```

A red arrow points from the word "CONTINUE;" in the SQL code to the number "5" in the output window.

Completion time: 2022-07-18T09:45:06.2174719+02:00

In het volgende voorbeeld wordt een tabel Numbers aangemaakt en gevuld met 1000 rijen met de waarden 1 t/m 1000 in de kolom n:

```

SET NOCOUNT ON;
DROP TABLE IF EXISTS Numbers;
CREATE TABLE Numbers(n INT NOT NULL PRIMARY KEY);
GO
DECLARE @i AS INT = 1;
WHILE @i <= 1000
BEGIN
    INSERT INTO Numbers (n) VALUES(@i);
    SET @i = @i + 1;
END;
SELECT * FROM Numbers;

```

SQLQuery16.sql - M...(USR\NIJP02 (56))\*

```

SET NOCOUNT ON;
DROP TABLE IF EXISTS Numbers;
CREATE TABLE Numbers(n INT NOT NULL PRIMARY KEY);
GO
DECLARE @i AS INT = 1;
WHILE @i <= 1000
BEGIN
    INSERT INTO Numbers (n) VALUES(@i);
    SET @i = @i + 1;
END;
SELECT * FROM Numbers;

```

179 %

n	
1	1
2	2
3	3
4	4
5	5
6	6
7	7
n	n

## 17.10 Cursors

Met een **Cursor** is het mogelijk om rijen één voor één in de gevraagde volgorde te verwerken uit een resultaat van een query. Cursoren dienen alleen gebruikt te worden als het niet anders kan daar cursoren neigen om het relationele model te verlaten en omdat cursoren veel trager zijn dan set-based code.

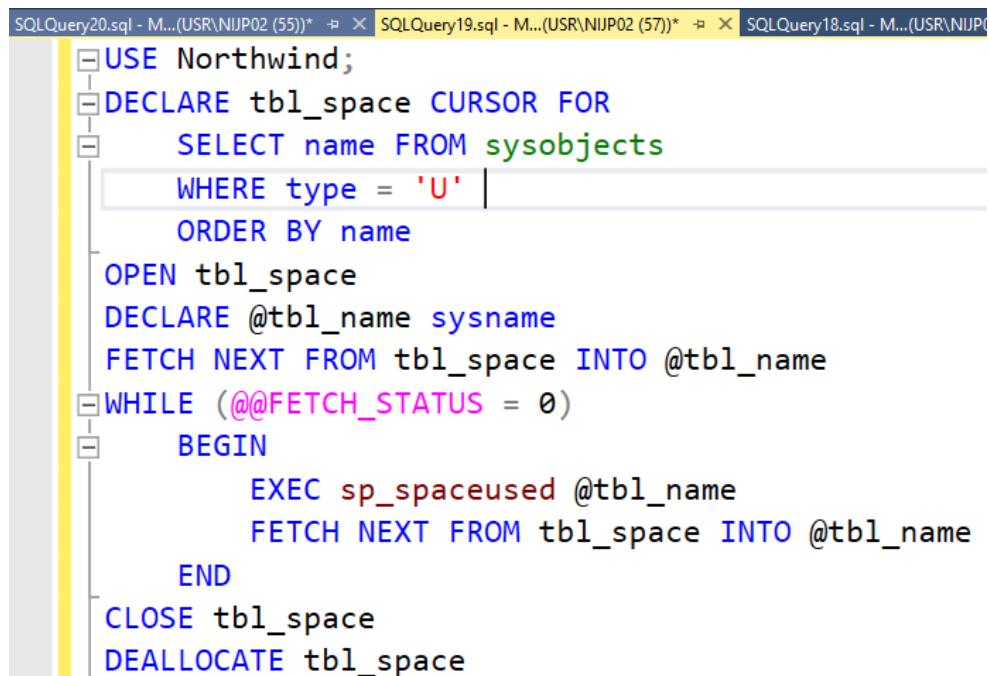
Werken met cursoren bestaat uit de volgende stappen:

1. Declareer de cursor gebaseerd op een query
2. Open de cursor
3. Fetch attribute waarden van de eerste cursor record naar de variabelen
4. Zolang je niet het einde van de cursor hebt bereikt loop door de cursor records. Voer in elke iteratie van de loop de verwerking uit die nodig is voor de huidige rij en fetch daarna de volgende attribute waarden van de volgende rij naar de variabelen
5. Sluit de cursor
6. Deallocate de cursor

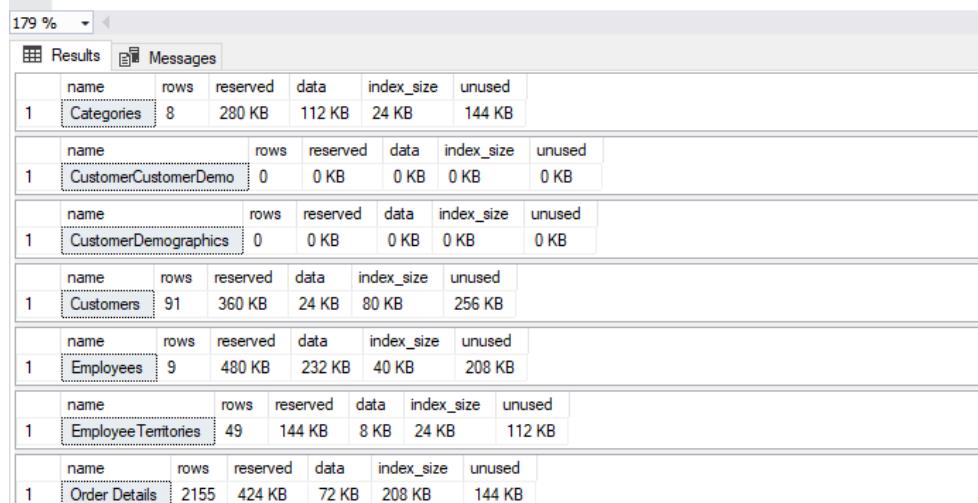
Voorbeeld:

Het onderstaande voorbeeld maakt een overzicht met alle USER objecten in de database Northwind, het aantal rijen in de tabellen en de gebruikte hoeveelheid opslagruimte door de objecten.

```
USE Northwind;
DECLARE tbl_space CURSOR FOR
    SELECT name FROM sysobjects
    WHERE type = 'U'
    ORDER BY name
OPEN tbl_space
DECLARE @tbl_name sysname
FETCH NEXT FROM tbl_space INTO @tbl_name
WHILE (@@FETCH_STATUS = 0)
    BEGIN
        EXEC sp_spaceused @tbl_name
        FETCH NEXT FROM tbl_space INTO @tbl_name
    END
CLOSE tbl_space
DEALLOCATE tbl_space
```



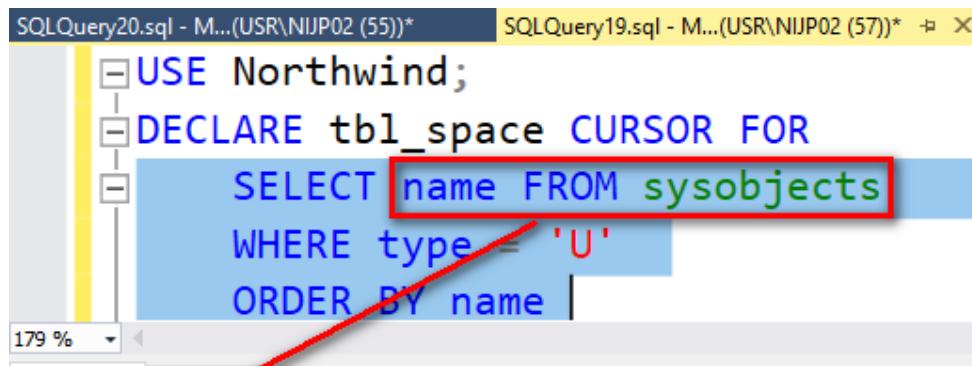
```
USE Northwind;
DECLARE tbl_space CURSOR FOR
    SELECT name FROM sysobjects
    WHERE type = 'U'
    ORDER BY name
OPEN tbl_space
DECLARE @tbl_name sysname
FETCH NEXT FROM tbl_space INTO @tbl_name
WHILE (@@FETCH_STATUS = 0)
    BEGIN
        EXEC sp_spaceused @tbl_name
        FETCH NEXT FROM tbl_space INTO @tbl_name
    END
CLOSE tbl_space
DEALLOCATE tbl_space
```



	name	rows	reserved	data	index_size	unused
1	Categories	8	280 KB	112 KB	24 KB	144 KB
1	CustomerCustomerDemo	0	0 KB	0 KB	0 KB	0 KB
1	CustomerDemographics	0	0 KB	0 KB	0 KB	0 KB
1	Customers	91	360 KB	24 KB	80 KB	256 KB
1	Employees	9	480 KB	232 KB	40 KB	208 KB
1	EmployeeTerritories	49	144 KB	8 KB	24 KB	112 KB
1	Order Details	2155	424 KB	72 KB	208 KB	144 KB

	name	rows	reserved	data	index_size	unused
1	Orders	830	776 KB	160 KB	320 KB	296 KB
1	Products	77	432 KB	8 KB	88 KB	336 KB
1	Region	4	144 KB	8 KB	24 KB	112 KB
1	Shippers	3	72 KB	8 KB	8 KB	56 KB
1	Suppliers	29	288 KB	24 KB	40 KB	224 KB
1	sysdiagrams	1	280 KB	96 KB	24 KB	160 KB
1	T1	0	0 KB	0 KB	0 KB	0 KB
1	Territories	53	144 KB	8 KB	24 KB	112 KB

De cursor `tbl_space` maakt gebruik van een **System Catalog View** `sysobjects`, die de namen van de User Defined objects (tabellen) weergeeft:



```

USE Northwind;
DECLARE tbl_space CURSOR FOR
    SELECT name FROM sysobjects
    WHERE type = 'U'
    ORDER BY name
  
```

The screenshot shows the SQL query being run in a query editor. A red arrow points from the word 'name' in the first `SELECT` statement to the results pane below. The results pane displays a list of table names:

	name
1	Categories
2	CustomerCustomerDemo
3	CustomerDemographics
4	Customers
5	Employees
6	EmployeeTerritories
7	Order Details
8	Orders
9	Products
10	Region
11	Shippers
12	Suppliers
13	sysdiagrams
14	T1
15	Territories

Na de declaratie wordt de cursor geopend en wordt er rij-voor-rij uit bovenstaande lijst gefetched. Telkens als de fetch een waarde heeft dan wordt m.b.v. de **System Stored Procedure** `sp_spaceused` aangeroepen om de opslagruimte van de tabellen op te vragen. De System Catalog Views en de System Stored Procedures worden in hoofdstuk 27 behandeld.

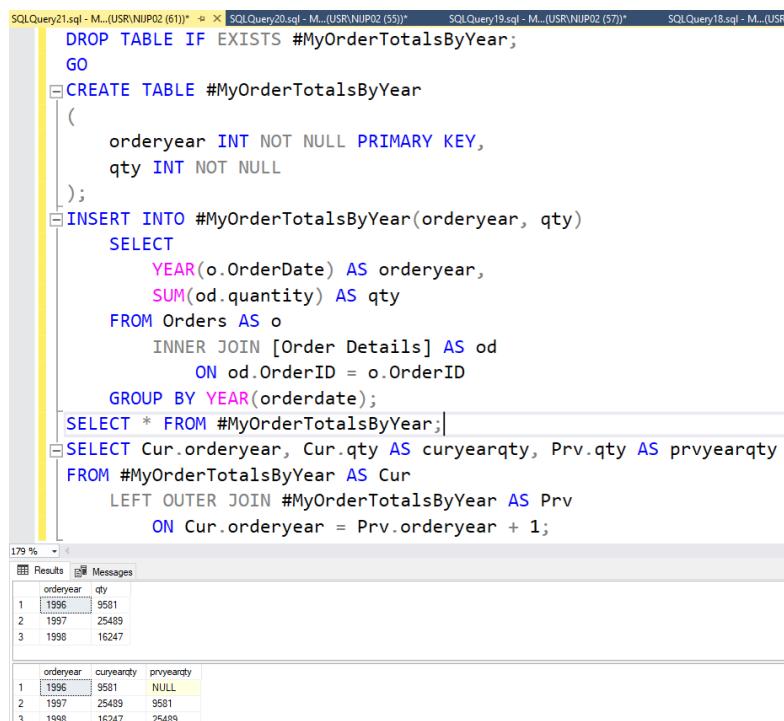
## 17.11 Temporary tables

Wanneer je tijdelijk gegevens in een tabel moet opslaan, dan heeft het vaak de voorkeur om geen permanente tabellen aan te maken. Bijvoorbeeld omdat je de gegevens alleen zichtbaar wil maken binnen de betreffende sessie of binnen de betreffende batch. In dat geval kan je Temporary Tables gebruiken.

**Temporary Tables** kunnen **lokaal** aangemaakt worden door het toevoegen van een prefix `#` en **globaal** (door iedereen te benaderen) door het toevoegen van een prefix `##`.

Voorbeeld:

```
DROP TABLE IF EXISTS #MyOrderTotalsByYear;
GO
CREATE TABLE #MyOrderTotalsByYear
(
    orderyear INT NOT NULL PRIMARY KEY,
    qty INT NOT NULL
);
INSERT INTO #MyOrderTotalsByYear(orderyear, qty)
SELECT
    YEAR(o.OrderDate) AS orderyear,
    SUM(od.quantity) AS qty
FROM Orders AS o
    INNER JOIN [Order Details] AS od
        ON od.OrderID = o.OrderID
    GROUP BY YEAR(orderdate);
SELECT * FROM #MyOrderTotalsByYear;
SELECT Cur.orderyear, Cur.qty AS curyearqty, Prv.qty AS prvyearqty
FROM #MyOrderTotalsByYear AS Cur
    LEFT OUTER JOIN #MyOrderTotalsByYear AS Prv
        ON Cur.orderyear = Prv.orderyear + 1;
```



The screenshot shows a SQL Server Management Studio window with four tabs at the top: SQLQuery21.sql - M... (USR\NJP02 (61)), SQLQuery20.sql - M... (USR\NJP02 (55)), SQLQuery19.sql - M... (USR\NJP02 (57)), and SQLQuery18.sql - M... (USR\NJP02 (57)). The main pane displays the T-SQL script for creating a temporary table and performing a self-join. Below the script, the 'Results' tab is selected, showing two result sets. The first result set shows the data inserted into the temporary table:

orderyear	qty
1996	9581
1997	25489
1998	16247

The second result set shows the output of the self-join query:

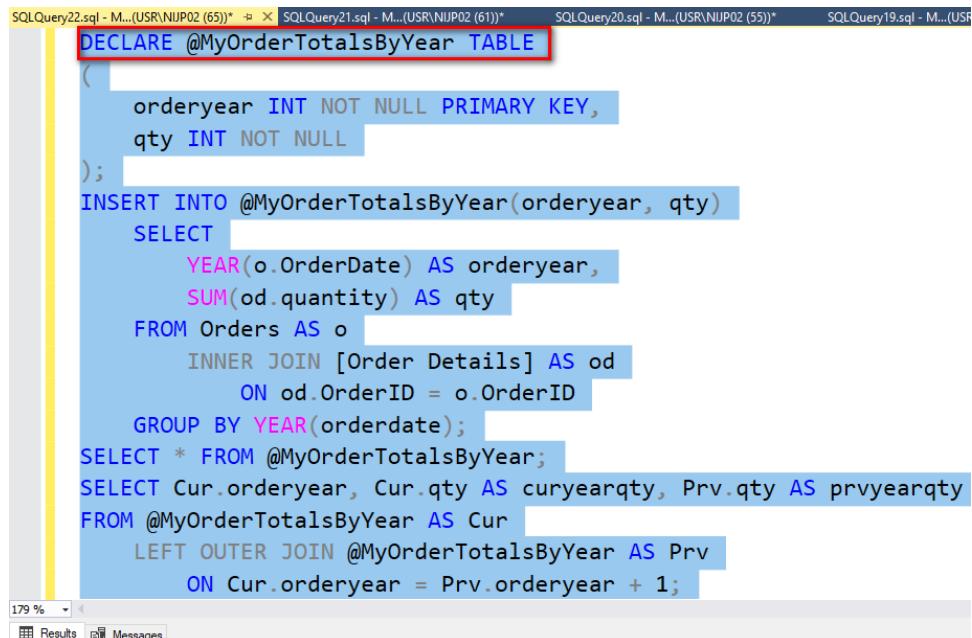
orderyear	curyearqty	prvyearqty
1996	9581	NULL
1997	25489	9581
1998	16247	25489

## 17.12 Table Variables

**Table Variables** zijn vergelijkbaar met Temporary Tables, maar een Table Variable wordt gedeclareerd zoals andere variabelen met **DECLARE** en niet met een **CREATE TABLE** statement. De prefix die bij een table variable wordt gebruikt is een **@**.

Als voorbeeld kunnen we hetzelfde voorbeeld nemen als in paragraaf 17.11 van een Temporary Table:

```
DROP TABLE IF EXISTS @MyOrderTotalsByYear;
GO
DECLARE @MyOrderTotalsByYear TABLE
(
    orderyear INT NOT NULL PRIMARY KEY,
    qty INT NOT NULL
);
INSERT INTO @MyOrderTotalsByYear(orderyear, qty)
SELECT
    YEAR(o.OrderDate) AS orderyear,
    SUM(od.quantity) AS qty
FROM Orders AS o
    INNER JOIN [Order Details] AS od
        ON od.OrderID = o.OrderID
    GROUP BY YEAR(orderdate);
SELECT * FROM @MyOrderTotalsByYear;
SELECT Cur.orderyear, Cur.qty AS curyearqty, Prv.qty AS prvyearqty
FROM @MyOrderTotalsByYear AS Cur
    LEFT OUTER JOIN @MyOrderTotalsByYear AS Prv
        ON Cur.orderyear = Prv.orderyear + 1;
```



orderyear	qty
1996	9581
1997	25489
1998	16247

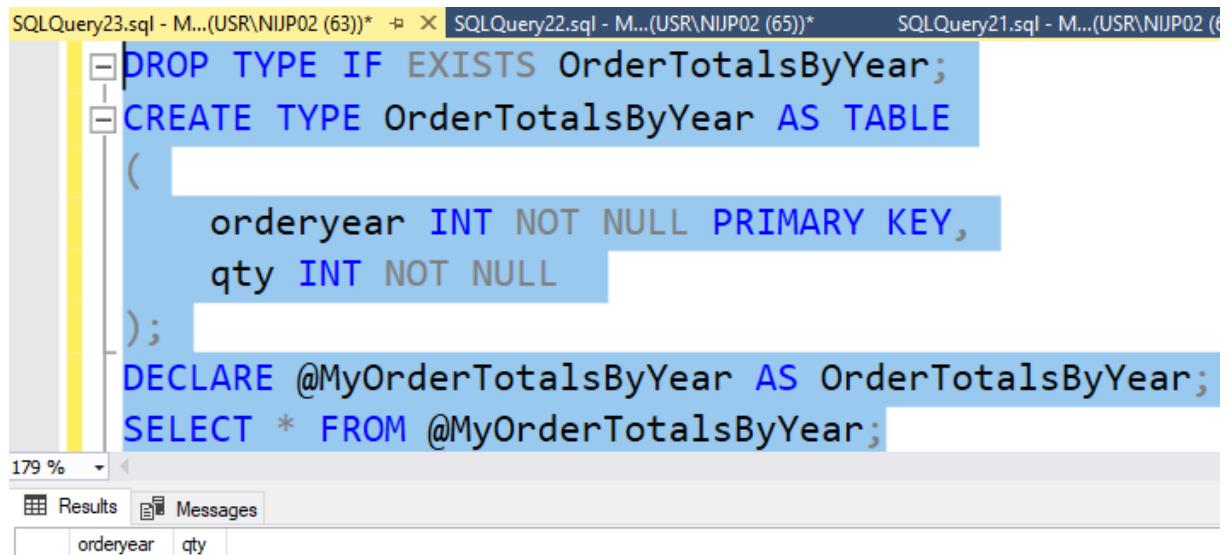
orderyear	curyearqty	prvyearqty
1996	9581	NULL
1997	25489	9581
1998	16247	25489

### 17.13 Table types

Een **Table Type** is te gebruiken om een tabel definitie te bewaren in een ander object in de database, zodat je het later kan hergebruiken.

Voornbeeld:

```
DROP TYPE IF EXISTS OrderTotalsByYear;
CREATE TYPE OrderTotalsByYear AS TABLE
(
    orderyear INT NOT NULL PRIMARY KEY,
    qty INT NOT NULL
);
DECLARE @MyOrderTotalsByYear AS OrderTotalsByYear;
SELECT * FROM @MyOrderTotalsByYear;
```



The screenshot shows a SQL query window with the following code:

```
DROP TYPE IF EXISTS OrderTotalsByYear;
CREATE TYPE OrderTotalsByYear AS TABLE
(
    orderyear INT NOT NULL PRIMARY KEY,
    qty INT NOT NULL
);
DECLARE @MyOrderTotalsByYear AS OrderTotalsByYear;
SELECT * FROM @MyOrderTotalsByYear;
```

The results pane shows a table structure:

orderyear	qty
-----------	-----

### 17.14 Dynamic SQL

Met SQL code kunnen we een batch maken en dan de batch executeren op SQL Server. Dit wordt Dynamic SQL genoemd. SQL Server biedt twee manieren om de batch op te starten **EXEC** en **sp\_executesql**, waarbij sp\_executesql veiliger en flexibeler is.

Voorbeeld EXEC:

```
DECLARE @sql AS VARCHAR(100);
SET @sql = 'PRINT ''Dit bericht was afgedrukt door een Dynamic SQL batch'';';
EXEC(@sql);
```

SQLQuery24.sql - M...(USR\NIJP02 (59))\*

```
DECLARE @sql AS VARCHAR(100);
SET @sql = 'PRINT ''Dit bericht was afgedrukt door een Dynamic SQL batch'';';
EXEC(@sql);
```

179 %

Messages

Dit bericht was afgedrukt door een Dynamic SQL batch

Completion time: 2022-07-18T22:04:11.8925896+02:00

Voorbeeld sp\_executesql:

```
DECLARE @sql AS NVARCHAR(100);
SET @sql = N'SELECT OrderID, CustomerID, EmployeeID, Orderdate
FROM Orders
WHERE OrderID = @orderid;';
EXEC sp_executesql
@tstmt = @sql,
@params = N'@orderid AS INT',
@orderid = 10248;
```

SQLQuery25.sql - M...(USR\NIJP02 (60))\*

```
DECLARE @sql AS NVARCHAR(100);
SET @sql = N'SELECT OrderID, CustomerID, EmployeeID, Orderdate
FROM Orders
WHERE OrderID = @orderid;';
EXEC sp_executesql
@tstmt = @sql,
@params = N'@orderid AS INT',
@orderid = 10248;
```

179 %

Results

	OrderID	CustomerID	EmployeeID	Orderdate
1	10248	VINET	5	1996-07-04 00:00:00.000

## Hoofdstuk 18 STORED PROCEDURES – (NEW)

### 18.1 Stored Procedures

Een **Stored Procedure** is een hoeveelheid code bestaande uit declaratieve en procedurele SQL instructies, die opgeslagen zijn in de catalog van de database en geactiveerd kan worden door deze Stored Procedure aan te roepen vanuit een programma, trigger of andere Stored Procedure.

Elke Stored Procedure bestaat uit minimaal drie delen: De naam van de Stored Procedure, een lijst met parameters en een body.

Voor de herkenbaarheid worden de namen van Stored Procedure vaak voorafgegaan door sp\_ waarbij sp aangeeft dat het een Stored Procedure is.

De lijst met parameters komt in het begin van de Stored Procedure definitie.

De BODY komt achter de AS en begint formeel met een BEGIN en eindigt met een END, echter mogen deze BEGIN en END in SQL Server ook weg gelaten worden. De body mag DDL, DCL en DML instructies bevatten zoals CREATE, GRANT, UPDATE, SELECT, etc. maar mag ook procedurele SQL instructies bevatten zoals IF- ELSE, WHILE, etc.

Bij het aanmaken van de Stored Procedure wordt er niets uitgevoerd, maar wordt er alleen gekeken of de syntax van de instructie in orde is en wordt de instructie opgeslagen in de catalog van de database. Het is dus te vergelijken met het creëren van een view.

Een Stored Procedure kan worden gemaakt met de syntax:

```
CREATE PROCEDURE <procedurenaam>
@<parameternaam> <datatype>
AS
  BEGIN
    <SQL code>
  END
GO
```

Een Stored Procedure kan aangeroepen worden met de syntax:

```
EXECUTE <procedurenaam> <waarde voor invoerparameters>
```

Een Stored Procedure kan worden verwijderd met de syntax:

```
DROP PROCEDURE <procedurenaam>;
```

Voorbeeld:

```
USE Northwind;
GO
drop procedure sp_GetEmployees;
CREATE PROCEDURE sp_GetEmployees
@FirstName nvarchar(10),
@LastName nvarchar(20)
AS
  BEGIN
```

```

SET NOCOUNT ON;
SELECT EmployeeID,TitleOfCourtesy,FirstName, LastName, Title, City
FROM Employees
WHERE FirstName = @FirstName AND LastName = @LastName;
END
GO

EXECUTE sp_GetEmployees N'Anne', N'Dodsworth';
GO
-- Or
EXEC sp_GetEmployees @FirstName = N'Anne', @LastName = N'Dodsworth';
GO

```

SQLQuery5.sql - MO...(USR\NIJP02 (59)) SQLQuery4.sql - MO...(USR\NIJP02 (52)) \* SQLQuery2.sql - MO...(USR\NIJP02 (51)) \* SQLQuery3.sql - MO...(USR\NIJP02 (54)) \*

```

USE Northwind;
GO
drop procedure sp_GetEmployees;
CREATE PROCEDURE sp_GetEmployees
@FirstName nvarchar(10),
@LastName nvarchar(20)
AS
BEGIN
    SET NOCOUNT ON;
    SELECT EmployeeID,TitleOfCourtesy,FirstName, LastName, Title, City
    FROM Employees
    WHERE FirstName = @FirstName AND LastName = @LastName;
END
GO

EXECUTE sp_GetEmployees N'Anne', N'Dodsworth';
GO
-- Or
EXEC sp_GetEmployees @FirstName = N'Anne', @LastName = N'Dodsworth';
GO

```

179 %

	EmployeeID	TitleOfCourtesy	FirstName	LastName	Title	City
1	9	Ms.	Anne	Dodsworth	Sales Representative	London

	EmployeeID	TitleOfCourtesy	FirstName	LastName	Title	City
1	9	Ms.	Anne	Dodsworth	Sales Representative	London

In bovenstaande Stored Procedure zijn INPUT parameters gebruikt. We kunnen ook OUTPUT parameters gebruiken.

Voorbeeld:

```

USE Northwind;
GO
drop procedure sp_EmployeesInCity;
CREATE PROCEDURE sp_EmployeesInCity
@City AS NVARCHAR(15),
@NumberEmployees AS INT OUTPUT
AS
BEGIN
    BEGIN

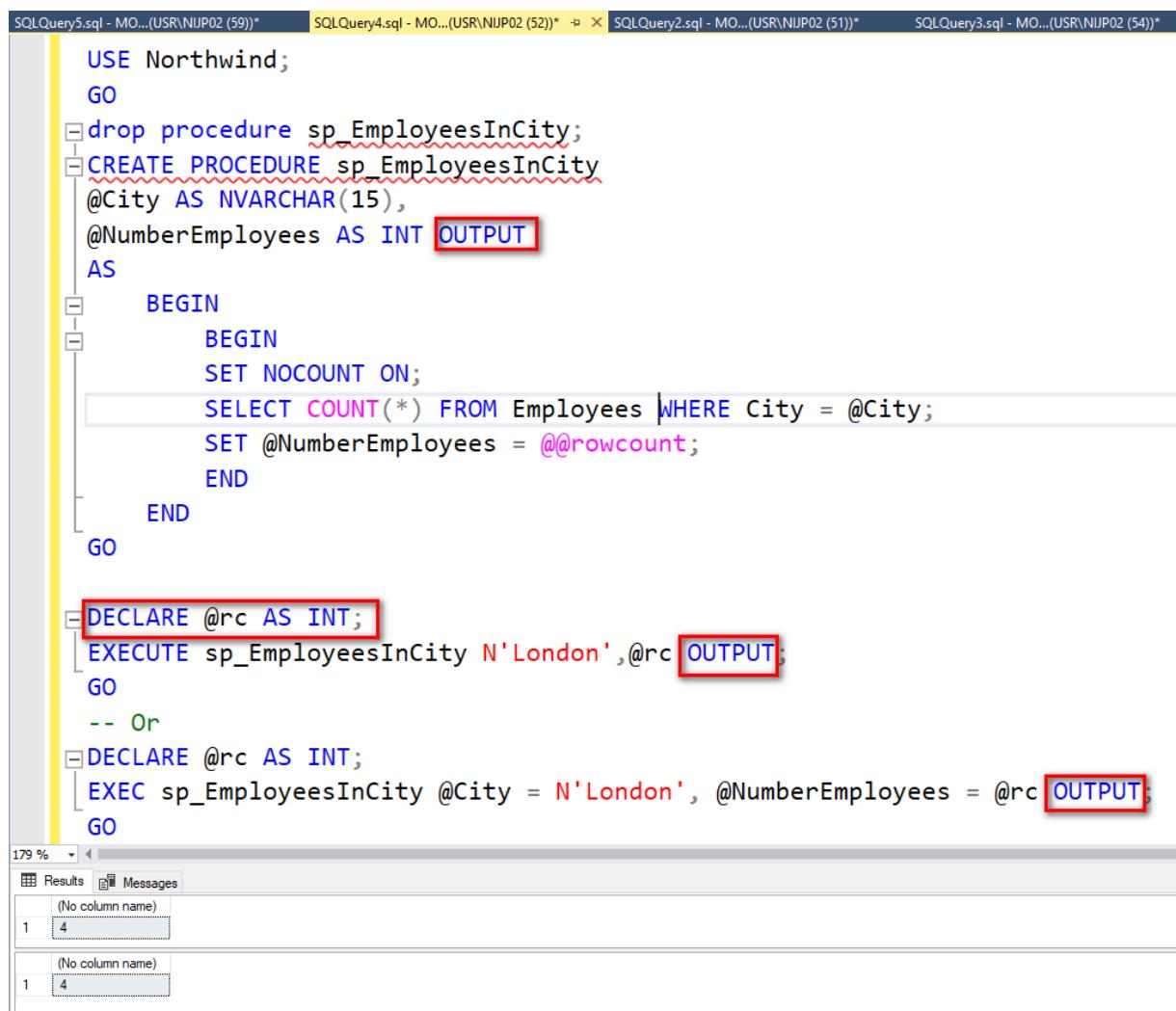
```

```

SET NOCOUNT ON;
SELECT COUNT(*) FROM Employees WHERE City = @City;
SET @NumberEmployees = @@rowcount;
END
GO

DECLARE @rc AS INT;
EXECUTE sp_EmployeesInCity N'London',@rc OUTPUT;
GO
-- Or
DECLARE @rc AS INT;
EXEC sp_EmployeesInCity @City = N'London', @NumberEmployees = @rc OUTPUT;
GO

```



The screenshot shows the SQL Server Management Studio interface with four tabs at the top: SQLQuery5.sql, SQLQuery4.sql, SQLQuery2.sql, and SQLQuery3.sql. The SQLQuery4.sql tab is active, displaying the following T-SQL code:

```

USE Northwind;
GO
drop procedure sp_EmployeesInCity;
CREATE PROCEDURE sp_EmployeesInCity
@City AS NVARCHAR(15),
@NumberEmployees AS INT OUTPUT
AS
BEGIN
    BEGIN
        SET NOCOUNT ON;
        SELECT COUNT(*) FROM Employees WHERE City = @City;
        SET @NumberEmployees = @@rowcount;
    END
END
GO

DECLARE @rc AS INT;
EXECUTE sp_EmployeesInCity N'London',@rc OUTPUT;
GO
-- Or
DECLARE @rc AS INT;
EXEC sp_EmployeesInCity @City = N'London', @NumberEmployees = @rc OUTPUT;
GO

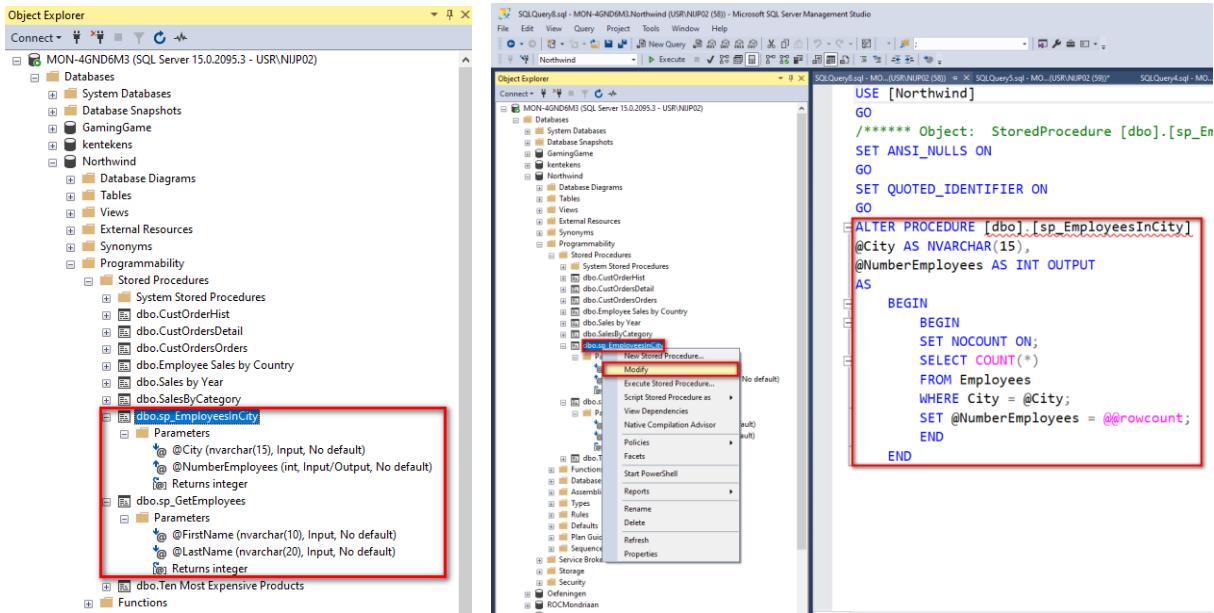
```

The code is annotated with red boxes highlighting the `OUTPUT` keyword in the parameter declaration and the `OUTPUT` clause in the `EXECUTE` statement. The results pane below shows two rows of data, both containing the value 4.

	(No column name)
1	4
1	4

In deze procedure wordt `@NumberEmployees` gedeclareerd als `OUTPUT` parameter door het woord `OUTPUT`. Omdat de Stored Procedure een waarde teruggeeft dient vóór de aanroep van de procedure eerst een variabele gedeclareerd te worden om de door de Stored Procedure teruggegeven waarde op te vangen (`@rc`).

De aangemaakte Stored Procedures zijn in SSMS terug te vinden in de databasesboom. Met rechtermuisklik -> Modify kan naar de code van de Stored Procedure worden gekeken en/of worden aangepast.



## Hoofdstuk 19 STORED FUNCTIONS - (NEW)

### 19.1 Stored Functions

**Stored Functions (User Defined Functions)** lijken erg op Stored Procedures. Het zijn ook stukken code bestaande uit SQL- en procedurele instructies die in de catalog opgeslagen worden en die vanuit programma's en SQL instructies aangeroepen kunnen worden.

De verschillen tussen Stored Functions en Stored Procedures zijn:

- Een Stored Function kan invoerparameters hebben maar geen uitvoerparameters. De Stored Function zelf is een uitvoerparameter.
- Stored Functions worden niet aangeroepen met een EXECUTE, maar kunnen net als de bekende scalaire functies binnen allerlei expressies aangeroepen worden.
- Stored Functions moeten een RETURN instructie bevatten, wat in een Stored Procedure niet toegestaan is.

Na de parameters volgt in de Stored Function definitie de RETURN specificatie, waarmee aangegeven wordt wat het datatype van de waarde is die de Stored Function weergeeft. Met de RETURN instructie geven we de Stored Function een waarde. Elke Stored Function moet minimaal één RETURN instructie bevatten. De RETURN instructie mag ook samengestelde expressies bevatten.

Als een Stored Function correct wordt verwerkt dan geeft de Stored Function een 1 (WAAR) als resultaat en anders de waarde 0 (ONWAAR).

Stored Functions mogen aangeroepen worden vanuit andere Stored Functions.

Een Stored Function kan aangemaakt worden met de syntax:

```
CREATE FUNCTION <functionnaam>
( <parameters> AS <datatype> )
RETURNS <datatype>
AS
BEGIN
    <SQL instructies>
END;
GO
```

Een Stored Function kan aangeroepen worden met de syntax:

```
SELECT <functionnaam> (<invoerparameters>) FROM <tabelnaam> ;
```

Een Stored Function kan verwijderd worden met de syntax:

```
DROP FUNCTION <functionnaam>;
```

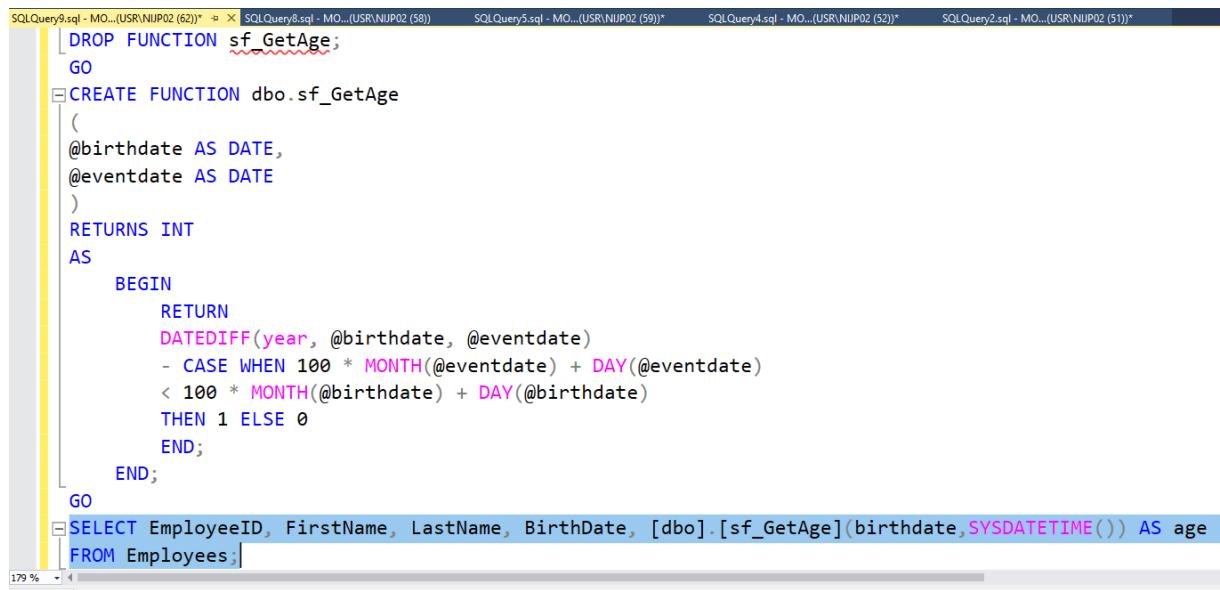
Voorbeeld:

```
USE Northwind;
DROP FUNCTION sf_GetAge;
GO
CREATE FUNCTION dbo.sf_GetAge
```

```

(
@birthdate AS DATE,
@eventdate AS DATE
)
RETURNS INT
AS
BEGIN
    RETURN
    DATEDIFF(year, @birthdate, @eventdate)
    - CASE WHEN 100 * MONTH(@eventdate) + DAY(@eventdate)
    < 100 * MONTH(@birthdate) + DAY(@birthdate)
    THEN 1 ELSE 0
    END;
END;
GO
SELECT EmployeeID, FirstName, LastName, BirthDate,
[dbo].[sf_GetAge](birthdate,SYSDATETIME()) AS age
FROM Employees;

```



The screenshot shows a SQL Server Management Studio window with multiple tabs open. The current tab displays the creation of a function named [dbo].[sf\_GetAge]. The code includes a function body that calculates age based on birthdate and eventdate, taking into account leap years. Below the function definition, a SELECT statement is shown, which uses the function to retrieve the age for all employees from the Employees table.

EmployeeID	FirstName	LastName	BirthDate	age
1	Nancy	Davolio	1948-12-08 00:00:00.000	73
2	Andrew	Fuller	1952-02-19 00:00:00.000	70
3	Janet	Leverling	1963-08-30 00:00:00.000	58
4	Margaret	Peacock	1937-09-19 00:00:00.000	84
5	Steven	Buchanan	1955-03-04 00:00:00.000	67
6	Michael	Suyama	1963-07-02 00:00:00.000	59
7	Robert	King	1960-05-29 00:00:00.000	62
8	Laura	Callahan	1958-01-09 00:00:00.000	64
9	Anne	Dodsworth	1966-01-27 00:00:00.000	56

## Hoofdstuk 20 TRIGGERS - (NEW)

### 20.1 Triggers

Een databaseserver is van nature passief en voert daarom pas een actie uit als we daar explicet met een SQL instructie opdracht toe geven. Door het gebruik van triggers kunnen we deze passieve databaseserver veranderen in een actieve databaseserver.

Een trigger is een hoeveelheid code, bestaande uit procedurele en declaratieve instructies, die opgeslagen zijn in de catalogus en door de databaseserver geactiveerd wordt als een bepaalde operatie op de database uitgevoerd wordt en alleen dan als een bepaalde conditie waar is.

Een trigger lijkt veel op een Stored Procedure, maar de wijze waarop een trigger aangeroepen wordt is anders dan bij de Stored Procedures. Triggers kunnen niet explicet aangeroepen worden, maar worden door SQL zelf aangeroepen zonder dat de programma's of gebruiker daar weet van hebben.

Een trigger wordt door SQL aangeroepen wanneer een programma/gebruiker/Stored Procedure een bepaalde database operatie uitvoert. Bijvoorbeeld wanneer een rij aan een tabel wordt toegevoegd, dan wordt de trigger automatisch door SQL uitgevoerd. Triggers kunnen we niet activeren of uitzetten.

Triggers kunnen we gebruiken voor het bijwerken van redundante gegevens en voor het bewaken van integriteit van de gegevens. Alle check-constraints zijn zeer eenvoudig met triggers te implementeren, maar daardoor kan de verwerkingssnelheid wel vertraagd worden. Daarom is het beter om integriteitsregels zoveel mogelijk met CHECK of Foreign Key constraints te implementeren en de oplossing m.b.v. triggers alleen te gebruiken als het niet anders kan.

Een trigger bestaat uit vier hoofdcomponenten: trigger-moment, trigger-event, trigger-conditie en trigger -actie.

Een trigger kan aangemaakt worden met de syntax:

```
CREATE TRIGGER <triggernaam>
<trigger-moment>
<trigger-event>
[<trigger-conditie>]
<trigger-actie>
```

Ofwel:

```
CREATE TRIGGER <triggernaam>
BEFORE|AFTER|INSTEAD OF
INSERT|DELETE|UPDATE [ OF <kolommenlijst>]
ON|OF|FROM|INTO <tabelnaam>
REFERENCING OLD|NEW|OLD_TABLE|NEW_TABLE
AS <tabelnaam>
FOR EACH ROW|STATEMENT
WHEN <conditie>
<being-end-blok>
```

Een trigger kan verwijderd worden met de syntax:

**DROP TRIGGER [<tabelnaam>.]<triggernaam>**

Voorbeeld:

In het onderstaand voorbeeld maken we een tabel T1, een audit tabel T1\_Audit en een trigger op tabel T1. Telkens wanneer we een INSERT uitvoeren op tabel T1 zal de trigger op de achtergrond daarvan een notitie maken in de T1\_Audit tabel. De gebruiker merkt of ziet er niets van.

Doel van de audit tabel T1\_Audit en de trigger is dat (voor audits) naderhand altijd terug te vinden is wie, wanneer, welke wijziging op de betreffende tabel heeft uitgevoerd. Een dergelijke constructie is bijvoorbeeld van belang op een salaris tabel. Wanneer een medewerker plotseling een gigantisch salaris ontvangt wil het management weten wie de wijziging daartoe in de tabel heeft gemaakt.

```
USE Oefeningen;
DROP TABLE dbo.T1, dbo.T1_Audit;
CREATE TABLE dbo.T1
(
keycol INT NOT NULL PRIMARY KEY,
datacol VARCHAR(10) NOT NULL
);
CREATE TABLE dbo.T1_Audit
(
audit_lsn INT NOT NULL IDENTITY PRIMARY KEY,
dt DATETIME2(3) NOT NULL DEFAULT (SYSDATETIME()),
login_name sysname NOT NULL DEFAULT(ORIGINAL_LOGIN()),
keycol INT NOT NULL,
datacol VARCHAR(10) NOT NULL
);
CREATE TRIGGER trg_T1_insert_audit ON dbo.T1 AFTER INSERT
AS
SET NOCOUNT ON;
INSERT INTO dbo.T1_Audit(keycol, datacol)
SELECT keycol, datacol FROM inserted;
GO
INSERT INTO dbo.T1(keycol,datacol) VALUES(10, 'a');
INSERT INTO dbo.T1(keycol,datacol) VALUES(30, 'x');
INSERT INTO dbo.T1(keycol,datacol) VALUES(20, 'g');

SELECT keycol, datacol FROM dbo.T1;
SELECT audit_lsn, dt, login_name, keycol, datacol
FROM dbo.T1_Audit;
```

SQLQuery10.sql - M...(USR\NIJP02 (57)) \* SQLQuery9.sql - MO...(USR\NIJP02 (62)) \* SQLQuery8.sql - MO...(USR\NIJP02 (58)) SQLQuery:

```

USE Oefeningen;
DROP TABLE dbo.T1, dbo.T1_Audit;
DROP TRIGGER trg_T1_insert_audit;
CREATE TABLE dbo.T1
(
keycol INT NOT NULL PRIMARY KEY,
datacol VARCHAR(10) NOT NULL
);
CREATE TABLE dbo.T1_Audit
(
audit_lsn INT NOT NULL IDENTITY PRIMARY KEY,
dt DATETIME2(3) NOT NULL DEFAULT (SYSDATETIME()),
login_name sysname NOT NULL DEFAULT(ORIGINAL_LOGIN()),
keycol INT NOT NULL,
datacol VARCHAR(10) NOT NULL
);
CREATE TRIGGER trg_T1_insert_audit ON dbo.T1 AFTER INSERT
AS
SET NOCOUNT ON;
INSERT INTO dbo.T1_Audit(keycol, datacol)
SELECT keycol, datacol FROM inserted;
GO

```

179 %

Messages

Commands completed successfully.

Completion time: 2022-07-16T21:25:14.5911645+02:00

SQLQuery10.sql - M...(USR\NIJP02 (57)) \* SQLQuery9.sql - MO...(USR\NIJP02 (62)) \* SQLQuery8.sql - MO...(USR\NIJP02 (58))

```

INSERT INTO dbo.T1(keycol,datacol) VALUES(10,'a');
INSERT INTO dbo.T1(keycol,datacol) VALUES(30,'x');
INSERT INTO dbo.T1(keycol,datacol) VALUES(20,'g');

SELECT keycol, datacol FROM dbo.T1;
SELECT audit_lsn, dt, login_name, keycol, datacol
FROM dbo.T1_Audit;

```

179 %

Results

	keycol	datacol
1	10	a
2	20	g
3	30	x

	audit_lsn	dt	login_name	keycol	datacol
1	1	2022-07-16 21:25:56.734	USR\NIJP02	10	a
2	2	2022-07-16 21:25:56.744	USR\NIJP02	30	x
3	3	2022-07-16 21:25:56.744	USR\NIJP02	20	g

## Hoofdstuk 21 DATABASEBEHEER

### 21.1 Taken/verantwoordelijkheden van de databasebeheerder

De databasebeheerder heeft verschillende taken en verantwoordelijkheden.

De meest belangrijke taken en verantwoordelijkheden zijn op volgorde van prioriteit:

1. Backup & Recovery
2. User Administration
3. Capacity Management
4. Performance Tuning

**Backup & Recovery:** De belangrijkste taak van de databasebeheerder is ervoor te zorgen dat er nooit data verloren zal gaan. De data in de database betreft meestal de data van het bedrijf waar het bedrijf voor de bedrijfsvoering van afhankelijk is. Als de data niet klopt of een deel van de data verloren is gegaan dan kan dat voor het bedrijf grote gevolgen hebben tot zelfs sluiting van het bedrijf.

**User Administration:** Nadat de databasebeheerder ervoor heeft gezorgd dat er nooit data verloren zal gaan dient de databasebeheerder ervoor te zorgen dat de juiste gebruiker bij de juiste gegevens kan komen in de database (niet meer en niet minder). Als een gebruiker niet bij de nodige gegevens kan komen dan kan de gebruiker zijn of haar werk niet uitvoeren. Wanneer een gebruiker echter bij te veel gegevens kan komen dan kan hij gegevens inzien of aanpassen waar hij of zij niet toe bevoegd is.

**Capacity Management:** Nadat de databasebeheerder ervoor heeft gezorgd dat er geen data uit de database verloren zal gaan en de juiste gebruikers bij de juiste data kan komen dient de databasebeheerder ervoor te zorgen dat de database bestanden en de schijven van de server nooit vol raken.

**Performance Tuning:** Nadat de databasebeheerder ervoor heeft gezorgd dat er geen data uit de database verloren zal gaan en de juiste gebruikers bij de juiste data kan komen en ervoor heeft gezorgd dat de database bestanden en de disks van de server niet vol kunnen raken dient de database beheerder ervoor te zorgen dat de database snel resultaat kan geven aan de gebruiker. In het algemeen kan worden gesteld dat hoe groter een database wordt, des te trager de database reageert. Om de database ondanks de grootte toch snel te laten reageren dient de databasebeheerder specifieke handelingen te verrichten.

In de praktijk zullen databasebeheerders bij vrijwel alle bedrijven deel moeten nemen aan een 24\*7 roulatie systeem waarbij de databasebeheerder om een aantal weken gedurende een week verantwoordelijk is voor de 24\*7 beschikbaarheid van de database. Hiervoor kan de databasebeheerder gedurende die week dag en nacht gebeld worden voor eventuele storingen en dient hij deze dan zo snel mogelijk op te lossen.

In de volgende hoofdstukken wordt een inleiding gegeven van deze taken en verantwoordelijkheden van de databasebeheerder.

## Hoofdstuk 22 BACKUP & RECOVERY

### 22.1 Backups

De belangrijkste taak van een databasebeheerder is het implementeren en onderhouden van een goede backup strategie. Database backups zijn essentieel voor elk bedrijf daar dataverlies grote gevolgen kunnen hebben voor het bedrijf.

Als voorbeeld van dataverlies kunnen we een fotocamera nemen. Wanneer iemand zijn fotocamera laat vallen, dan kan de lens kapot gaan. De lens kan erg duur zijn, maar het is te vervangen. Wanneer echter de memory card in de fotocamera defect raakt en de foto's verloren zijn gegaan dan is het niet meer mogelijk om de herinneringen op de gemaakte foto's terug te halen als er geen goede backup van is gemaakt. De verloren foto's zijn dus niet te vervangen!

Ditzelfde geldt voor informatiesystemen. Wanneer een computeronderdeel, server of netwerkcomponent defect raakt kan dat erg duur zijn, maar het is te vervangen. Wanneer er echter data in de database verloren is gegaan dan is de data niet te vervangen en kan dat voor het bedrijf een zodanig groot probleem zijn dat het bedrijf daardoor zal moeten sluiten.

Dataverlies bij een bank kan bijvoorbeeld betekenen dat de bank niet meer weet hoeveel geld de klanten op hun rekeningen hebben staan of hoeveel schulden de klanten hebben. In dat geval kan de bank niet meer goed functioneren en kan het tot sluiting van de bank leiden.

De databasebeheerder moet zich daarom altijd bewust zijn dat hij werkt met de data waarvan het bedrijf afhankelijk is. Een databasebeheerder mag vele fouten maken, maar de grootste fout die een databasebeheerder kan maken is dataverlies. Het maken en terugzetten van backups lijkt eenvoudig, maar kan ook behoorlijk complex worden. Vaak denkt een databasebeheerder dat er dagelijks goede backups worden gemaakt zonder te testen of de backups daadwerkelijk terug te zetten zijn. Net zoals het noodzakelijk is dat EHBO-mensen periodiek oefenen in het reanimeren is het voor databasebeheerders noodzakelijk om te oefenen in het terugzetten van backups naar een specifiek punt in tijd (Point in Time Recovery).

In dit hoofdstuk behandelen we een introductie in het maken en terugzetten van database backups. De uitleg van database backups en database recovery is in dit hoofdstuk sterk versimpeld en vereenvoudigd.

#### 22.1.1 Recovery Models

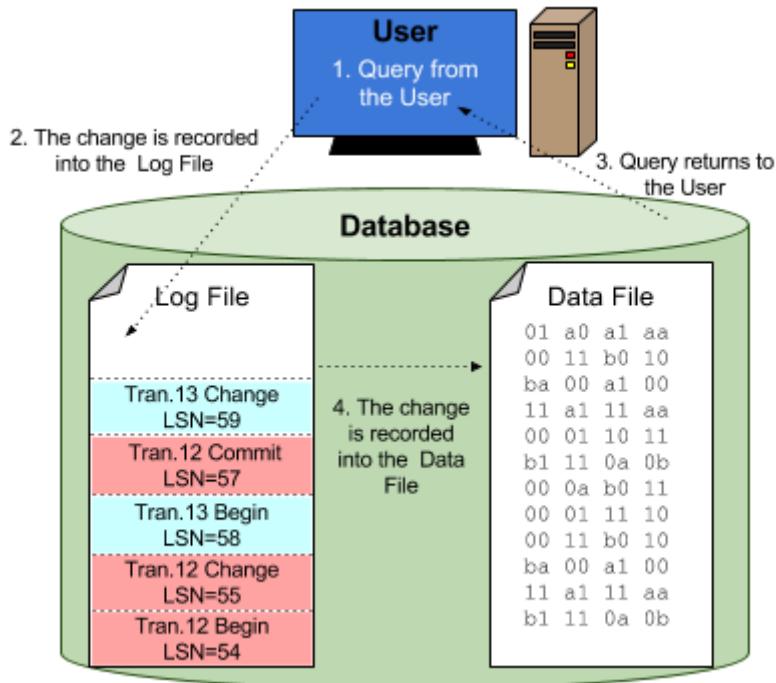
##### Simple Recovery Model

In een klein bedrijf waarbij de database alleen overdag wordt gebruikt en dataverlies van de dag zelf acceptabel is voldoet een simpele backup strategie. Er kan dan elke avond na werktijd een backup van de database worden gemaakt en bij een probleem kan de backup van de vorige avond worden terug gezet. De eventuele wijzigingen in de database nadat de backup is gemaakt is daarmee echter wel verloren gegaan.

##### Full Recovery Model

Bij de meeste bedrijven kunnen tegenwoordig echter 24/7 wijzigingen in de database worden gemaakt en is het niet acceptabel dat er terug wordt gegaan naar de backup van de avond ervoor, daar er op die dag zelf ook belangrijke wijzigingen in de database zijn gemaakt die niet verloren mogen gaan.

In dat geval hebben we een backup strategie nodig waarmee we in staat zijn om ook de wijzigingen die na de backup zijn gemaakt terug te halen. Hiervoor wordt een transaction log geïntroduceerd.



### 22.1.2 Transaction log

Een **transaction log** (Log File) is een relatief klein bestand waarin alle wijzigingen (INSERT, UPDATE, CREATE, ALTER, etc) in de database beknopt en met een tijdstempel (hoe laat het gebeurd was) worden beschreven. De stappen in de bovenstaande figuur zijn:

1. De gebruiker maakt een wijziging in de database m.b.v. een SQL query
2. De SQL query wordt direct uitgevoerd en de wijzigingen worden opgeslagen in de transaction log
3. Zodra de wijziging in de transaction log is opgeslagen komt de query terug bij de gebruiker met de melding dat de wijziging is uitgevoerd
4. Het grote database bestand (Data File) waarin de tabellen daadwerkelijk zijn opgeslagen wordt pas later aangepast gedurende een zogenaamde **Checkpoint**.

Zodra een wijziging in de transaction log is opgeslagen wordt dus aan de gebruiker gemeld dat de wijziging is opgeslagen. Daarna wordt de wijziging ook nog geschreven in de tabellen welke zijn opgeslagen in de database bestanden middels het checkpoint. Het schrijven naar het relatief kleine transaction log bestand is aanzienlijk sneller dan het schrijven naar de tabellen in de grote database bestanden. Het RDBMS kan hiermee dus snel antwoord geven aan de gebruiker, nog voordat de checkpoint is uitgevoerd. De gebruiker kan dus van het RDBMS een melding krijgen dat de wijziging opgeslagen is voordat de wijziging daadwerkelijk in de tabellen staat.

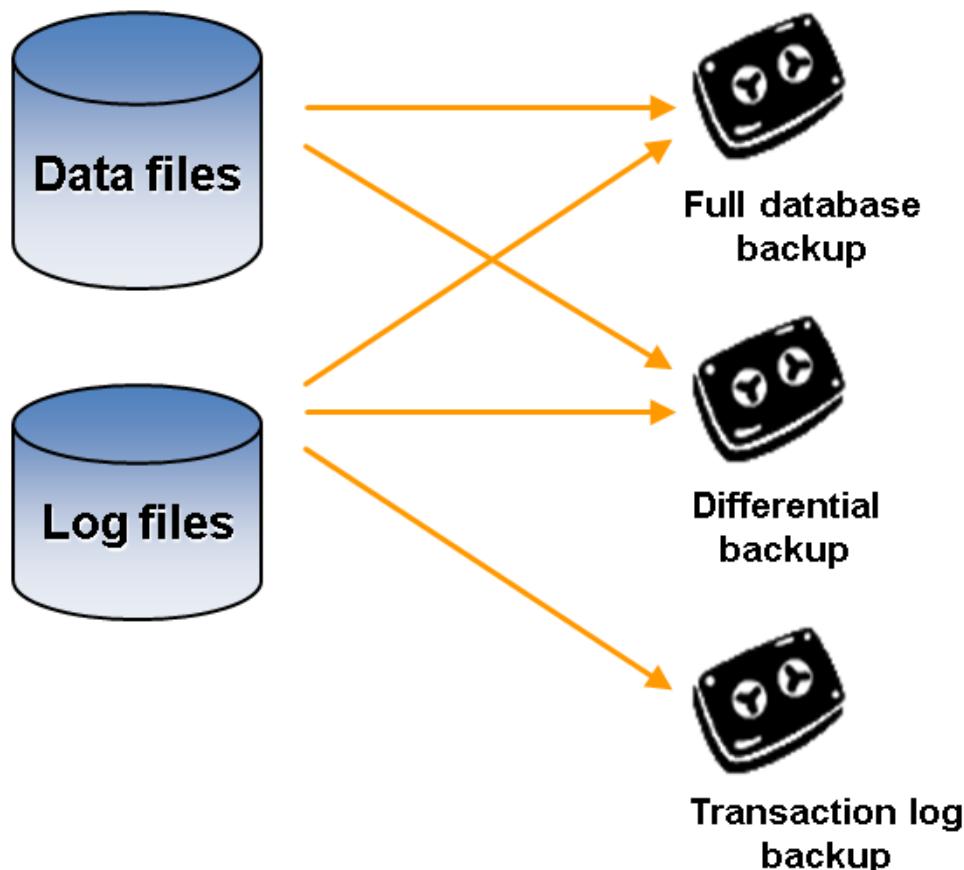
Met behulp van alle wijzigingen in de transaction log kunnen we na het terughalen van de backup van de avond ervoor (Restore) de database terug brengen naar de situatie die er was na de backup (Recovery). We kunnen zelfs de database hiermee terug brengen naar elk gewenst punt in tijd (Point in Time Recovery). Zo kunnen we bijvoorbeeld na een menselijke fout waarbij een tabel per ongeluk is verwijderd de database terug brengen naar het moment vlak voordat de tabel was verwijderd.

### 22.1.3 Full Backup, Incremental Backup en Differential backup

De tabellen in de database bestanden (Data File) worden tijdens een database backup gekopieerd naar een Backup Systeem (bijvoorbeeld naar tape of naar een disk op een andere server). Deze database bestanden zijn vaak erg groot (Terabytes) en een backup van deze database bestanden kan vele uren duren. Het is zelfs mogelijk dat de database zodanig groot is dat het niet mogelijk is om binnen 24 uur een volledige backup (Full Backup) van de database te maken.

De grote database bestanden bevatten voornamelijk tabellen met ongewijzigde data. Slechts een klein deel van de data wordt per dag gewijzigd. Het is daarom overbodig om dagelijks van alle data in de database bestanden een backup te maken. Het voldoet om dagelijks slechts een backup te maken van de tabellen waarin op die dag daadwerkelijk wijzigingen zijn gemaakt.

Daarom worden er naast de volledige backups (**Full Backup**) ook Incremental en Differential backups gemaakt. Een **Differential backup** is een backup van de tabellen die gewijzigd zijn sinds de laatste Full Backup. Een **Incremental backup** is een backup van de tabellen die gewijzigd zijn sinds de laatste Incremental of Full backup).



In de praktijk wordt veelal één keer in de week een Full Backup gemaakt in het weekend omdat er dan slechts weinig gebruikers aan het werk zijn met de database. Vervolgens wordt elke dag een Differential dan wel een Incremental Backup gemaakt om dagelijks een backup te maken van de gewijzigde tabellen. Zo'n dagelijkse Incremental of Differential Backup duurt aanzienlijk korter dan de wekelijkse Full backup. Bij de keuze voor een backup strategie moet er naast de Full backup gekozen worden voor het gebruik van Differential Backups, dan wel Incremental Backups. Het is niet mogelijk om Differential Backups en Incremental Backups door elkaar te gebruiken.

Naast de database backups (wekelijkse Full Backup en dagelijkse Differential- of Incremental Backup) moet er ook periodiek een backup worden gemaakt van het transaction log bestand. Bij kritische systemen kan dit elke minuut zijn, terwijl dit bij minder kritische systemen mogelijk een halve dag is. De databasebeheerder dient zich er enerzijds van bewust te zijn dat de acties in de transaction log verloren kunnen gaan zolang er geen backup van is gemaakt. Anderzijds kan het te frequent maken van transaction log backups de reactietijd (Response Time) van de database aan de gebruikers vertragen. Het is voor de databasebeheerder dus zoeken naar de juiste balans tussen veilige transaction log backups en de reactietijd van het RDBMS. Telkens wanneer er een backup is gemaakt van het transaction log bestand wordt de inhoud van het transaction log bestand leeg gemaakt en wordt het weer opnieuw gevuld.

#### 22.1.4 Point in Time Recovery

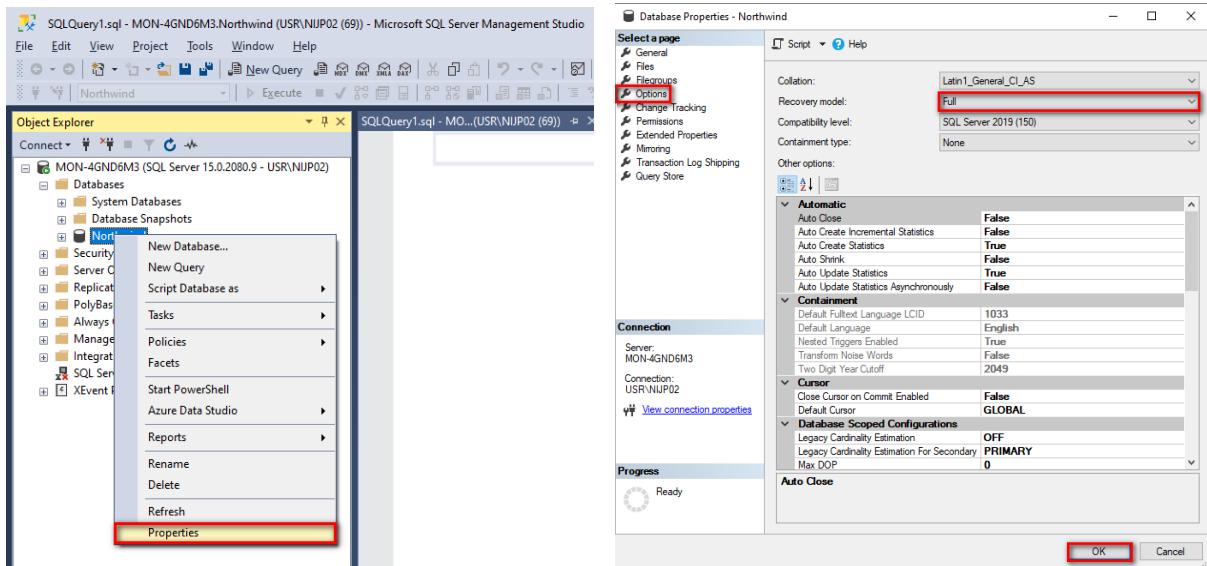
Bij het terughalen van een backup naar een specifiek punt in tijd dient de volgende stappen te worden uitgevoerd:

1. Er wordt een backup gemaakt van het aanwezige transaction log bestand indien deze backup nog niet was gemaakt
2. Full backup van het laatste weekend wordt terug gehaald van het Backup Systeem en overschrijft de oude database bestanden
3. In geval van Differential Backup:  
Laatste Differential Backup van gisterenavond wordt terug gehaald van het Backup Systeem en overschrijven de oude database bestanden  
In geval van Incremental Backups:  
Alle Incremental Backups na de Full Backup tot en met gisterenavond wordt terug gehaald van het Backup Systeem en overschrijven de oude database bestanden
4. Transaction Log backups sinds de backup van gisterenavond worden terug gehaald van het Backup Systeem
5. M.b.v. de teruggehaalde transaction log backups worden alle acties in de transaction log (vanaf de laatste full-, incremental- of differential backup) opnieuw uitgevoerd tot het gewenste punt in tijd (rollforward). Indien er op het punt in tijd transactions aanwezig waren die nog niet afgerond waren dan worden deze transactions teruggedraaid (rollback)

#### 22.1.5 SQL Server Full Recovery Model

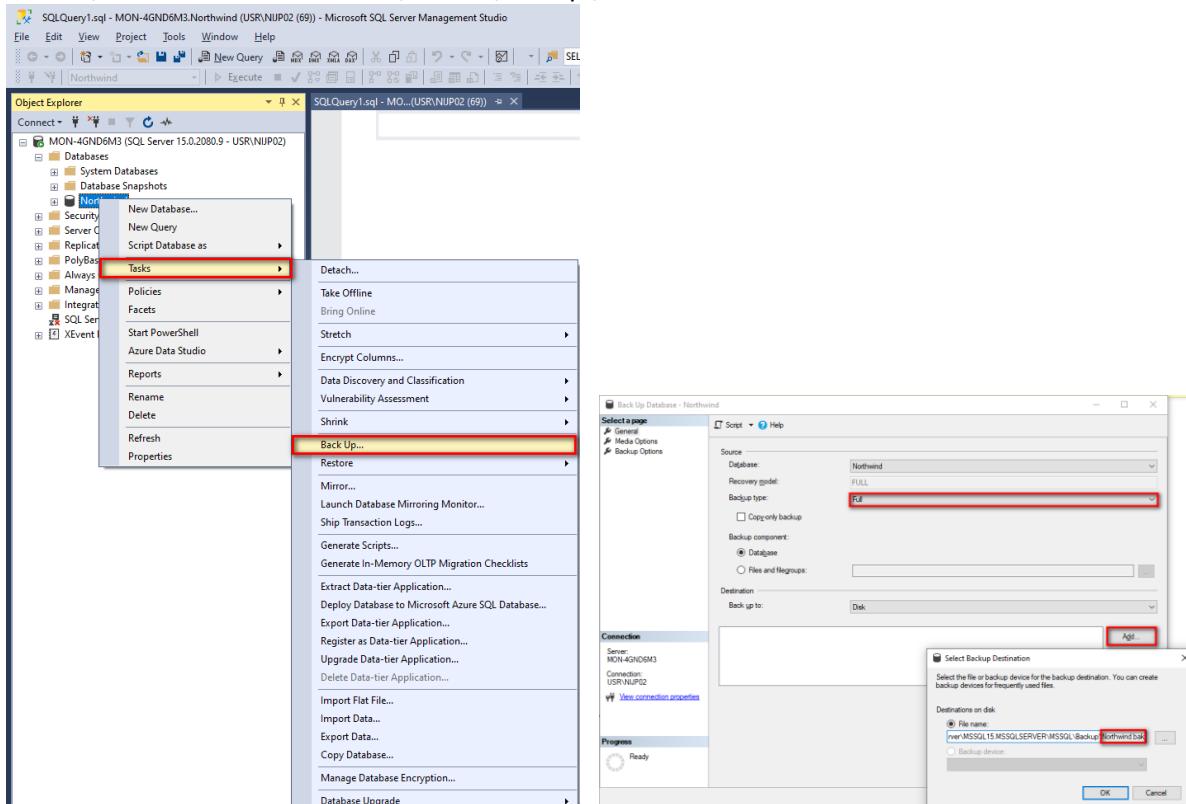
In Microsoft SQL Server worden géén Incremental Backups ondersteund en wordt gebruik gemaakt van Differential Backups.

Het Recovery Model kan in SQL Server worden aangepast door rechtermuisklik op database Northwind -> Properties -> Options -> Recovery Model = Full -> OK



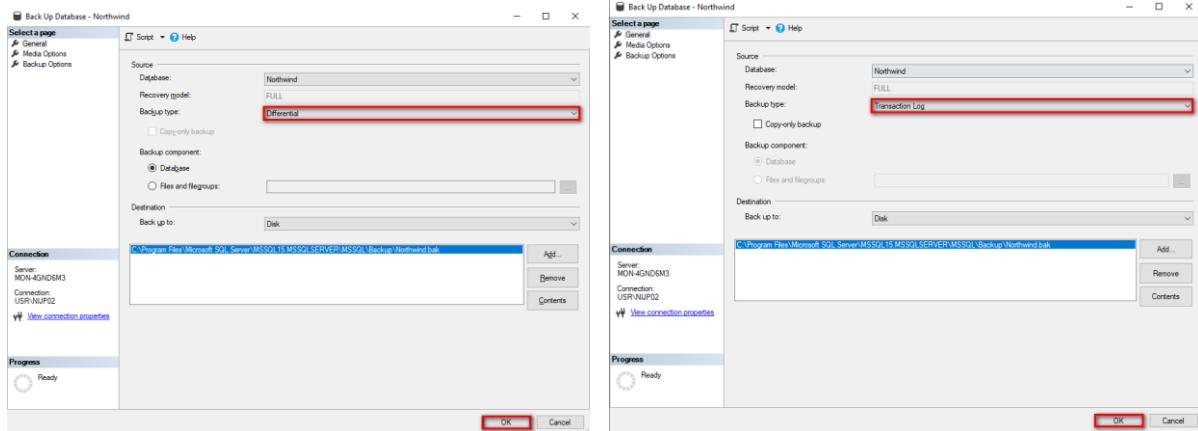
## 22.2 Backup

Daarna kan een Full Backup worden gemaakt door rechtermuisklik op database Northwind -> Tasks -> Back Up... -> Backup Type = Full -> Add -> File name = C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\MSSQL\Backup\Northwind.bak -> OK

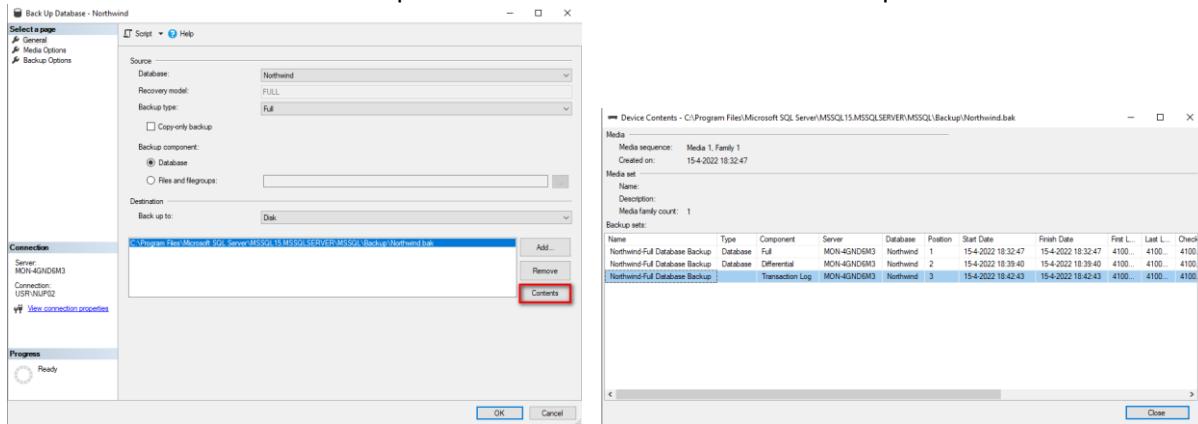


Er dient altijd eerst een Full backup te worden gemaakt voordat er Differential en/of Transaction Log backups kunnen worden gemaakt.

Nadat er een Full backup is gemaakt kan een Differential en/of een Transaction Log backup worden gemaakt door rechtermuisklik op database Northwind -> Tasks -> Back Up... -> Backup Type = Differential (of Transaction Log) -> OK

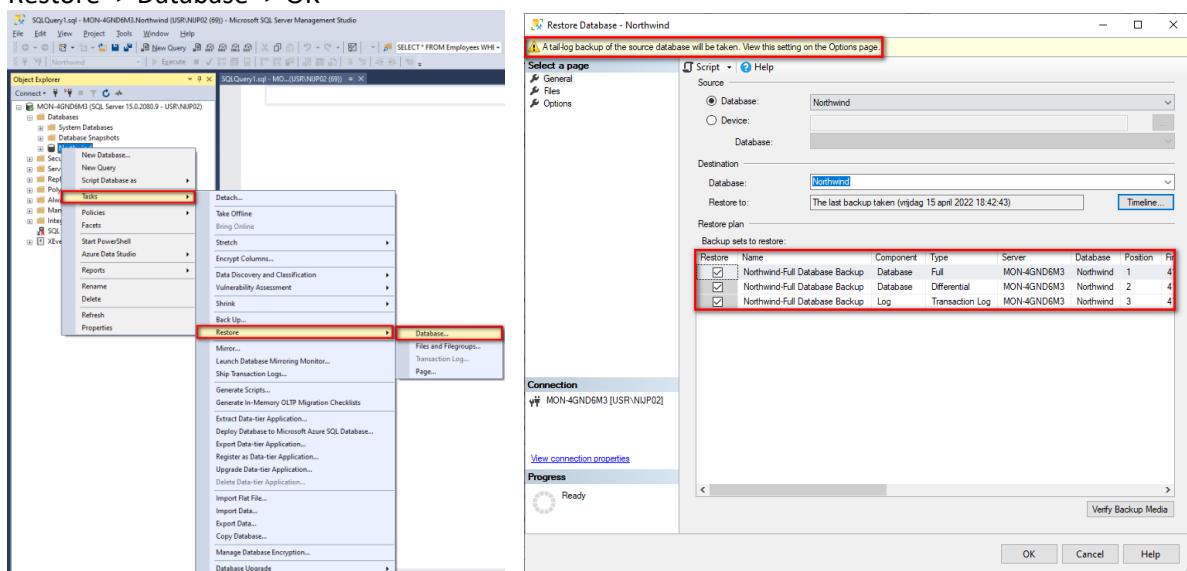


Alle gemaakte backups zijn nu opgeslagen in hetzelfde backup bestand Northwind.bak en is zichtbaar te maken door rechtermuisklik op database Northwind -> Tasks -> Back Up... -> Contents



## 22.3 Recovery

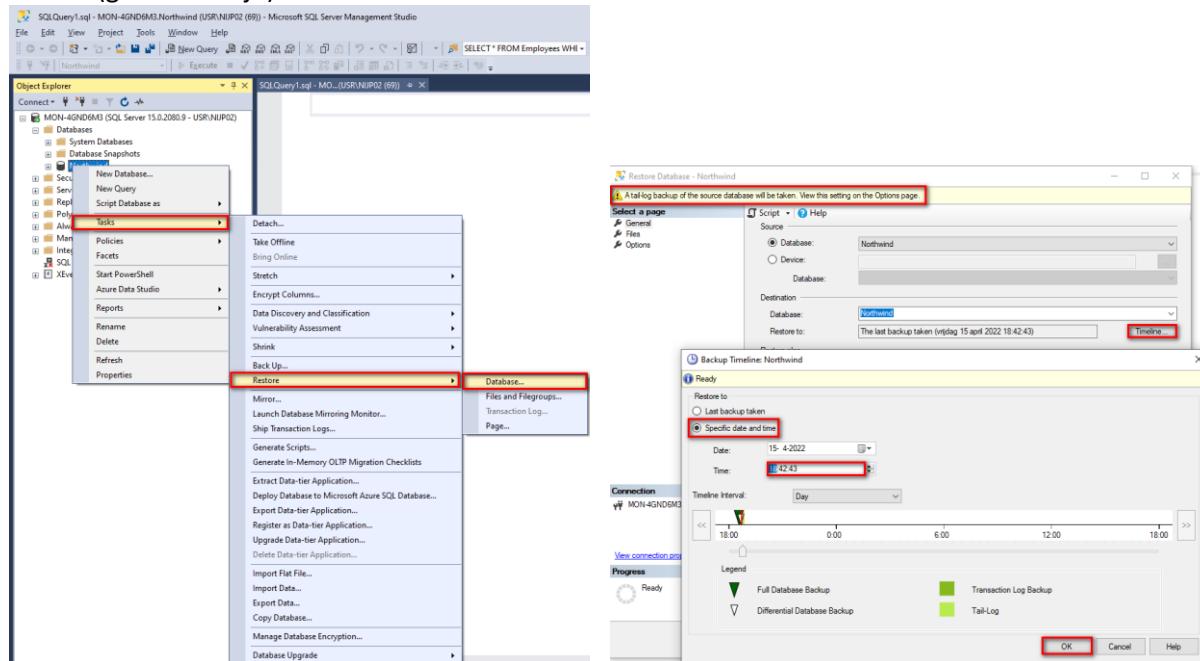
Een database backup is terug te halen door rechtermuisklik op database Northwind -> Tasks -> Restore -> Database -> OK



## 22.4 Tail-log backup

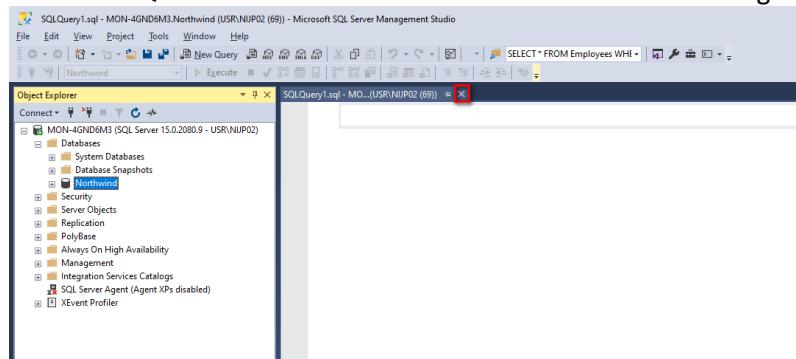
Default wordt bij een Restore eerst een backup gemaakt van alle wijzigingen in het Transaction Log (tail-log backup) bestand zodat er een backup is van alle uitgevoerde transacties en er geen transacties in de transaction log verloren zullen gaan (zie bovenin het scherm de tekst “A tail-log backup of the source database will be taken. View the setting on the Options page.”). Vervolgens wordt de database terug gezet naar de laatste status van de database, dus tot en met de laatste wijziging die op de database was uitgevoerd.

Een database backup terug halen naar een specifiek punt in tijd (Point in Time) kan door rechtermuisklik op database Northwind -> Tasks -> Restore -> Database -> Specific date and time -> Time = (gewenste tijd) -> OK



Ook hierbij wordt weer eerst een backup gemaakt van alle wijzigingen in het Transaction Log bestand (zie bovenin het scherm de tekst “A tail-log backup of the source database will be taken. View the setting on the Options page.”). Vervolgens kan een gewenst tijdstip binnen de backups worden geselecteerd waar de database naartoe moet worden hersteld (Recovery).

Een database backup kan echter niet worden terug gezet zolang er nog gebruikers in de database aanwezig zijn, inclusief jouw eigen sessie. Verbreek daarom eerst alle verbindingen met de database door alle SQL Windows te sluiten voordat de Restore wordt uitgevoerd:



Een volledige database Restore wordt bijvoorbeeld terug gezet wanneer een systeem is gecrashed of een disk defect is geraakt en de database moet worden hersteld (recovered) tot de laatste wijziging in de database.

Een Point in Time Recovery wordt bijvoorbeeld uitgevoerd wanneer er een menselijke fout is geweest zoals per ongeluk een tabel verwijderd of leeg gemaakt. In dat geval kan bv. eerst m.b.v. het uitlezen van audit files gezocht worden naar het tijdstip dat de fout heeft plaats gevonden en dan kan de database hersteld (recovered) worden tot kort voor dat tijdstip dat de fout is opgetreden.

Microsoft documentatie over Backup en Restore zie

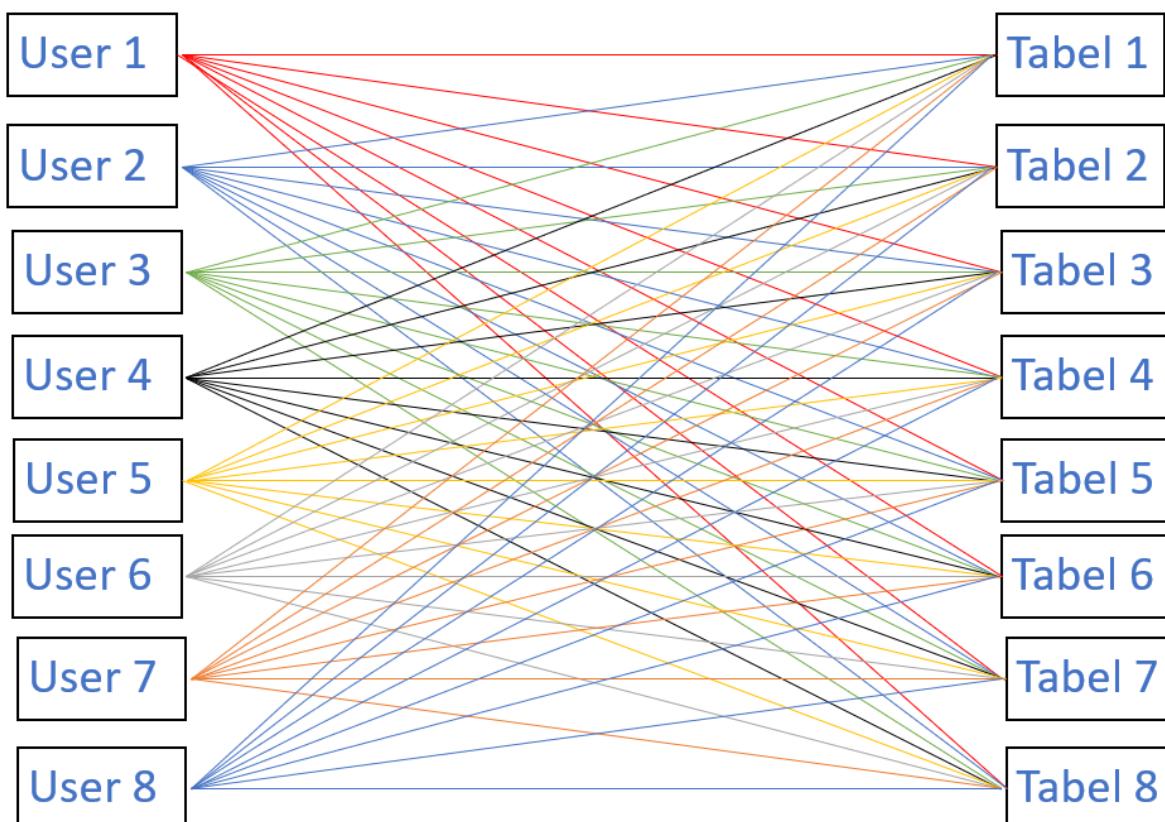
<https://docs.microsoft.com/en-us/sql/relational-databases/backup-restore/back-up-and-restore-of-sql-server-databases?view=sql-server-ver15>

## Hoofdstuk 23 USER ADMINISTRATION – TODO

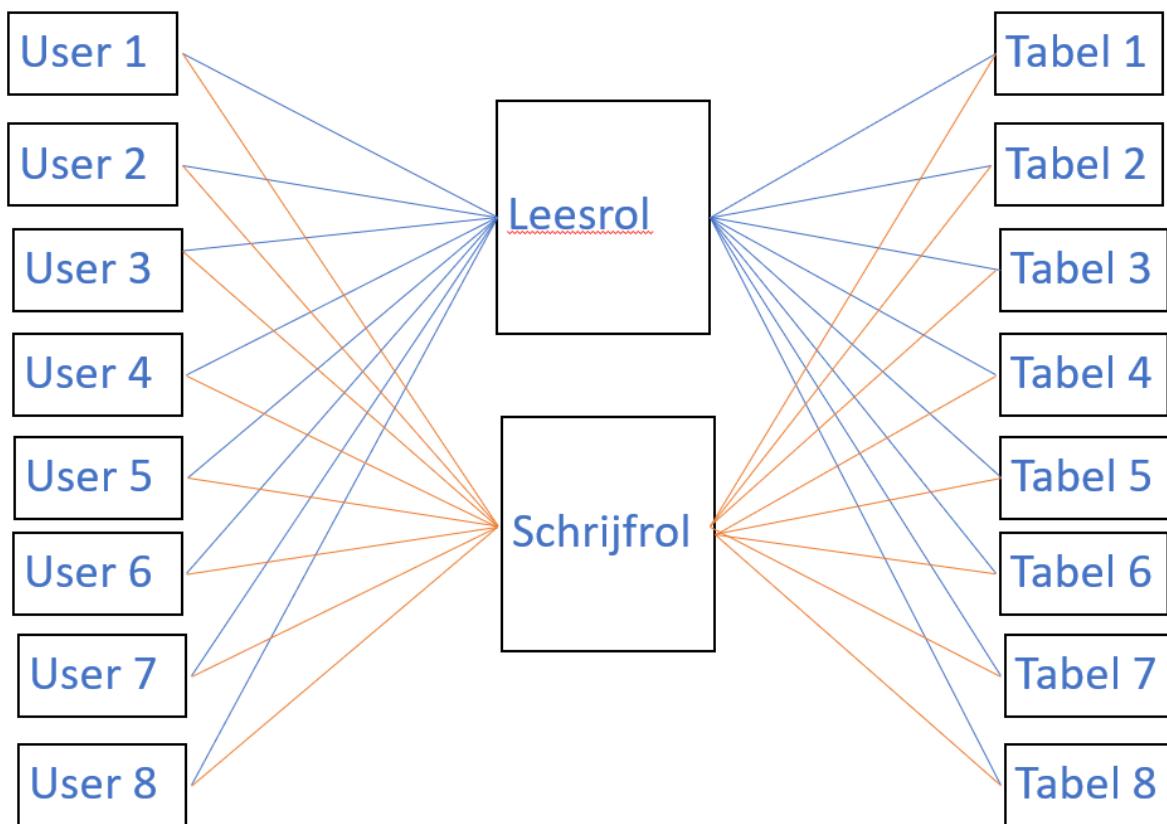
### 23.1 User Administration

In een database hebben we verschillende objecten zoals tabellen, indexen, views, functions, stored procedures, etc. en we hebben verschillende soorten gebruikers zoals medewerkers van de Human Resource afdeling, medewerkers van de verkoopafdeling, medewerkers van de inkoopafdeling, medewerkers van het magazijn, etc. Elke afdeling maakt gebruik van een specifiek aantal tabellen en de gegevens in deze tabellen zijn vaak niet bedoeld om in te zien of gewijzigd te worden door medewerkers van andere afdelingen. Zo wil de Human Resource afdeling niet dat medewerkers van andere afdelingen in de tabellen kunnen kijken of kunnen wijzigen waarin de adressen of de salarissen staan van alle medewerkers van het bedrijf. In dat geval zouden medewerkers van andere afdelingen bijvoorbeeld het salaris van zichzelf kunnen verhogen of het salaris van de directeur kunnen verlagen.

Om ervoor te zorgen dat de juiste personen bij de juiste gegevens kan komen (en niet meer en niet minder dan dat) kunnen we per object expliciete permissies per gebruiker toekennen. We kunnen elke medewerker van de Human Resource afdeling SELECT, INSERT en UPDATE permissies geven op de tabellen met de adressen en de salarissen van alle medewerkers. We kunnen elke medewerker van de afdeling Communicatie SELECT rechten geven op de tabel met de adres gegevens van alle medewerkers zonder rechten op de tabel met de salarissen van alle medewerkers. Op deze manier krijgen we een soort van spinnenweb met allerlei permissies tussen objecten en gebruikers, en wordt het erg onoverzichtelijk:



Door het gebruik van Roles kunnen we het beheer van gebruikers en permissies aanzienlijk vereenvoudigen. We kunnen zo Roles aanmaken voor medewerkers van een bepaalde afdeling of nog beter per functie. Zo kunnen we een Role maken voor de medewerkers van de Human Resource afdeling die zich daadwerkelijk bezig houden met de salarisafhandelingen zonder dat andere medewerkers van de Human Resource afdeling bij de gegevens met de salariën kunnen komen. Een gebruiker krijgt een minimaal aantal Roles toegewezen die hij of zij nodig heeft voor het uitvoeren van zijn/haar werkzaamheden. En elke Role krijgt SELECT, INSERT of UPDATE permissies op de nodige tabellen, waardoor het spinnenweb met permissies tussen objecten en gebruikers aanzienlijk overzichtelijker en eenvoudiger te beheren wordt:



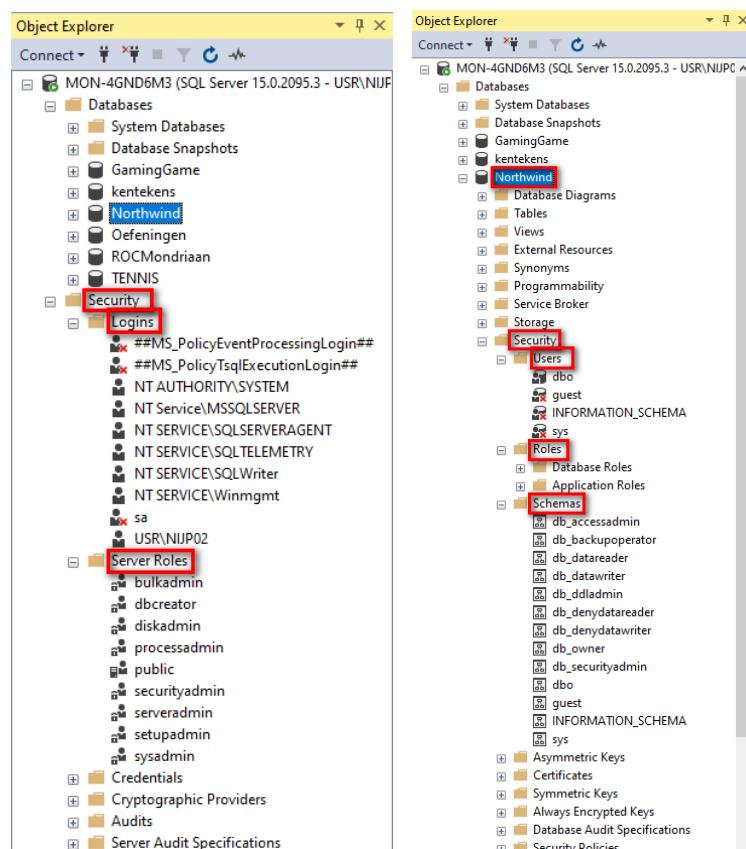
In de praktijk worden permissies vrijwel uitsluitend aan gebruikers toegekend via Roles en worden expliciete permissies direct aan gebruikers alleen in uitzonderlijke situaties toegekend. In het algemeen worden er per set tabellen minimaal twee Roles toegekend, een Leesrol (SELECT) en een schrijfrol (INSERT, UPDATE) zodat er onderscheid kan worden gemaakt tussen de gebruikers die de gegevens mogen lezen en de gebruikers die de gegevens mogen toevoegen of mogen wijzigen. Let op dat wanneer een gebruiker wel een schrijfrol maar niet een leesrol op een set tabellen krijgt, dat hij/zij dan wel de gegevens kan wijzigen maar niet meer kan terugzien. Er kunnen functionele redenen zijn om nog meer Roles toe te voegen, bijvoorbeeld per specifieke functie. Zo heeft een IT auditor vaak op veel tabellen van verschillende afdelingen leesrechten nodig om zijn werk als auditor te kunnen uitvoeren. De User Administrator die de permissies aan de gebruikers toekent mag pas toegang aan tabellen van een bepaalde afdeling toekennen nadat hij/zij daar explicite toestemming voor heeft gekregen van de manager van de betreffende afdeling. De User Administrator is zelf immers niet in staat om te kunnen beoordelen wie er bijvoorbeeld lees- of schrijfrechten mogen

hebben tot de salarisgegevens. Zelfs voor de databasebeheerder zelf moet het onmogelijk zijn om ongestraft aan salarisgegevens te komen.

De databasebeheerder is er verantwoordelijk voor dat alle permissies in de database correct zijn. Het komt in de praktijk voor dat een databasebeheerder binnen een project een script van een ontwikkelaar ontvangt om bepaalde wijzigingen in de database te maken. De databasebeheerder dient zich dan bewust te zijn dat hij/zij verantwoordelijk blijft voor de permissies in de database en dient dan ook alle permissies in het script te controleren of deze wel voldoen aan de regels van het bedrijf. Achteraf kan de databasebeheerder de verantwoordelijkheid van de onjuiste permissies niet meer afschuiven op de ontwikkelaar, de databasebeheerder blijft verantwoordelijk en zal dat tijdens audits moeten kunnen verantwoorden!

## 23.2 Logins, Users, Server Roles, Database Roles, Owners, schemas

SQL Server kent Logins, Server Roles, Users, Database Roles, Owners en Schemas.



In SSMS zijn Logins en Server Roles zichtbaar onder Security van SQL Server en gelden voor de gehele database server DBMS.

Users, Roles en Schemas zijn zichtbaar binnen elke database en gelden alleen binnen de betreffende database specifiek.

## 23.3 Logins

Om in SQL Server te kunnen komen heeft een gebruiker een **Login** nodig, maar daarmee kan de gebruiker nog niet in een database komen. Om in een database te kunnen komen heeft de gebruiker naast de Login op SQL Server ook nog een **User** nodig in de betreffende database die gekoppeld is

aan de Login. Vaak is de naam van de User identiek aan de naam van de Login, maar dat hoeft niet het geval te zijn.

### 23.4 Server Roles

Aan een Login kunnen we verschillende **Server Roles** toekennen, afhankelijk van de soort gebruiker. Deze Server Roles hebben diverse permissies op de Database Engine, het database management system, waarmee bijvoorbeeld de database kan worden down gebracht of de databasebestanden kan worden uitgebreid of database geheugeninstellingen kan worden aangepast.

Zo kan aan een databasebeheerder bijvoorbeeld de Server Role **sysadmin** worden toegekend, waarmee hij/zij toegang krijgt tot alle gegevens in alle databases en tot alle parameters van de Database Engine.

Server Roles zijn iets anders dan Database Roles. Binnen de databases kunnen **Database Roles** of directe expliciete **permissies** op tabellen aan Users worden toegekend.

### 23.5 Owners

Elke tabel heeft een **Owner** (eigenaar). Meestal is dit de User die de CREATE TABLE instructie heeft laten verwerken.

Namen van tabellen zijn uniek binnen een Owner (eigenaar). Met andere woorden, twee Users mogen allebei een tabel creëren met dezelfde naam omdat ze elk de Owner zijn van hun eigen tabel, maar een User mag niet dezelfde naam toekennen aan twee van zijn tabellen.

Als een User een tabel van een andere Owner wil benaderen, dan moet deze precies aangeven welke tabel hij bedoelt, door de Owner van de te benaderen tabel vóór de tabelnaam in de tabelspecificatie op te nemen.

Bijvoorbeeld:

```
SELECT * FROM dbo.Customers;
```

Hierbij is dbo (database owner) de naam van de Owner van de tabel Customers.

Dit is niet verplicht voor het gebruik van een eigen tabel, maar het mag ook wel gebruikt worden voor een verwijzing naar een eigen tabel.

### 23.6 Owners en Schemas

Naast de Owner kent SQL Server ook nog een Schema. Een database is een object om tabellen fysiek te groeperen, zoals een opbergkast voor mappen. Een schema is daartegen een andere methode om database objecten zoals tabellen en views te groeperen. Met een schema kunnen objecten logisch gegroepeerd worden. Opbergmappen kunnen bijvoorbeeld logisch gegroepeerd worden door er speciale labels op te plakken. Zoals alle mappen met een label waarop staat 'Inkoop' die documenten bevatten van de inkoopafdeling.

Wanneer een User een tabel voor zichzelf maakt, dan noemen we de User de **Creator** van de tabel en wordt hij automatisch ook de Owner van de tabel. Wanneer de User echter een tabel aanmaakt voor een andere User dan wordt de User de Creator maar de andere User de Owner van de tabel.

Bijvoorbeeld:

```
CREATE TABLE dbo.Test (kolom1 int);
```

Voor de tabelnaam staat de naam van de eigenaar dbo, gescheiden door een punt. Als deze naam niet wordt gespecificeerd dan is de tabel eigendom van de bouwer.

In feite zijn de termen Owner en Schema gelijkwaardig. Alle databaseobjecten van dezelfde eigenaar behoren tot hetzelfde Schema. De naam van het Schema is gelijk aan de naam van de Owner. Dus met de vorige CREATE TABLE instructie wordt een tabel gecreëerd die aan het schema dbo wordt toegekend.

Het zijn niet alleen tabellen die tot een schema kunnen behoren. Ook indexen, views, Stored Procedures, etc. kunnen tot een schema behoren.

Tevens geldt dat tabellen behorende tot hetzelfde schema één database kunnen vormen.

~~In het bovenstaande voorbeeld CREATE TABLE dbo.Test (kolom1 int); werd een schema impliciet gecreëerd.~~

We kunnen een nieuw Login maken met het statement:

```
CREATE LOGIN <loginnaam> WITH PASSWORD = '<paswoord>';
```

We kunnen een nieuwe User aanmaken met het statement:

```
USE <databasenaam>
CREATE USER <usernaam> FOR LOGIN <loginnaam> WITH DEFAULT_SCHEMA <schemanaam>;
GO;
```

We kunnen een nieuw Schema aanmaken met het statement:

```
CREATE SCHEMA <schemanaam>;
```

Na het uitvoeren van dit statement bestaat het schema, maar bevat het nog geen database objecten.

Met het CREATE SCHEMA instructie kunnen we ook diverse objecten tegelijkertijd creëren die tot het nieuwe schema behoren. Voordeel hiervan is dat het allemaal verwerkt wordt of allemaal niet. Het is dus echt alles of niets. Bij een DROP SCHEMA worden alle objecten van het schema eveneens verwijderd. Als de DROP instructie wordt uitgebreid met RESTRICT dan wordt de instructie afgeweerd als er nog object in het schema zitten.

Er bestaan volgende verschillen tussen Schema en User:

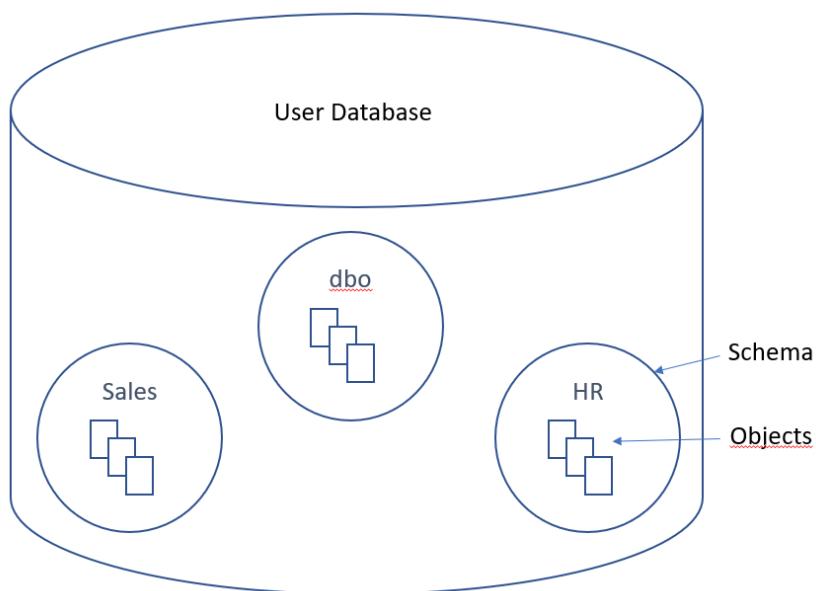
- Een schema heeft geen wachtwoord, terwijl voor een User, dus ook een eigenaar, wel een wachtwoord gedefinieerd kan worden
- Met een schemanaam kun je niet inloggen
- Met de GRANT instructie kun je geen bevoegdheden aan schema's toekennen maar wel aan eigenaren
- Een schema kan geen bouwer zijn maar een User wel
- Met de CREATE SCHEMA instructie kan met 1 instructie een verzameling databaseobjecten gecreëerd worden, maar met een CREATE USER instructie niet

In feite komt het er op neer dat Schema en User overlappende functionaliteiten hebben, maar elk heeft ook unieke mogelijkheden.

Definitie van Microsoft: A schema is distinct namespace to facilitate the separation, management and ownership of database objects

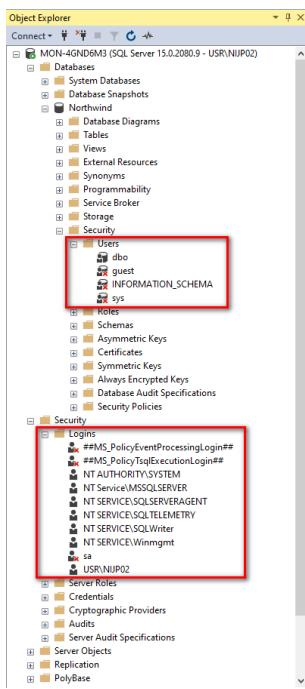
We kunnen schema's vergelijken met kamers in een woning. In elke kamer staat een bed en een stoel. Één van de kamers is de default kamer. Als ik aan iemand vraag om de stoel te halen dan is onduidelijk uit welke kamer de stoel moet worden gehaald, maar omdat ik een default kamer gedefinieerd heb dan zal de stoel uit de default kamer worden gehaald. Als ik vraag om de stoel uit kamer 1 te halen dan zal de stoel uit kamer 1 worden gehaald.

Een schema is een container met objecten zoals tabellen, views, stored procedures etc. Je kan permissies beheersen op Schema niveau. Je kan bv een User GRANT SELECT toekennen op een Schema zodat de User alle tabellen binnen het Schema kan opvragen. Dus security is een van de redenen om objecten in schema's te arrangeren. Het schema is ook een namespace – prefix voor het objectnaam. Microsoft adviseert om in de code altijd two-part object namen te gebruiken. Met een three part object name kan je ook de database aangeven en met een four-part name kan je ook een andere instance aangeven.



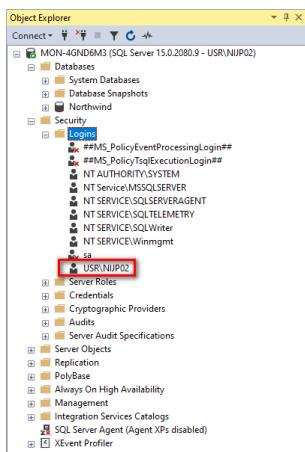
## 23.7 Logins

SQL Server kent Logins en Users. Logins zijn nodig om de SQL Server binnen te komen en Users zijn nodig om een specifieke database binnen te komen.

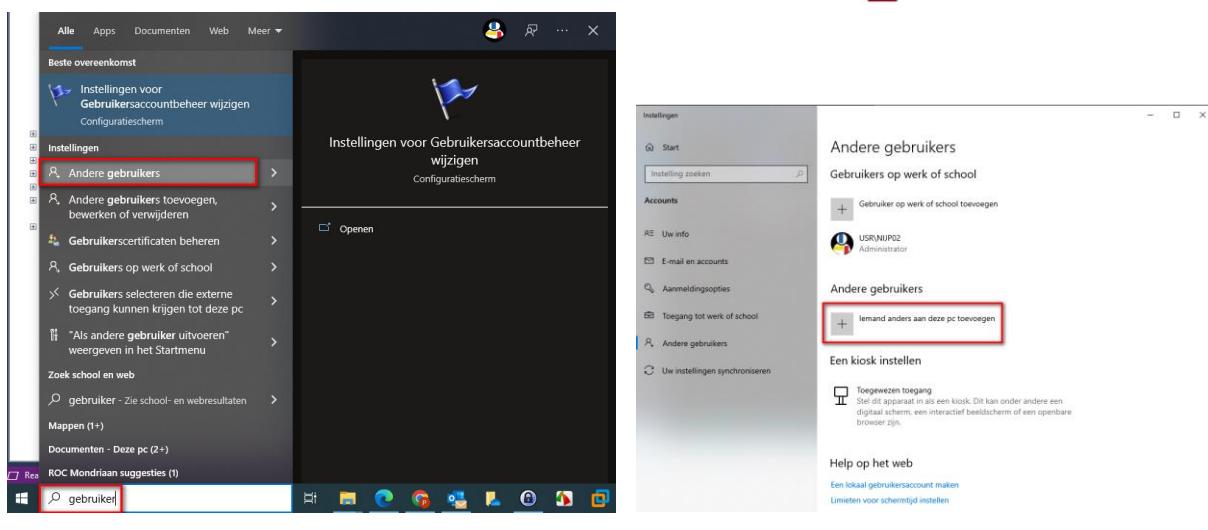


Het is te vergelijken met een stad in de 18<sup>e</sup> eeuw met daaromheen een stadsmuur. De hele stad is omringd door al de muren er om heen. Om de stad binnen te komen heb je een Login nodig, die jou toelaat om door de poort te gaan. Maar hoewel je de stad binnen bent gekomen kan je nog niet in de huizen komen. Om in de huizen te komen (in dit geval de database) heb je een User nodig die toegewezen (mapped) is aan jouw Login. Dus je hebt eerst een Login nodig om in SQL Server te komen en dan een User om in een specifieke database te komen. Het heeft niet zo veel zin als je de stad binnen bent gekomen maar geen gebouw in kan.

Omdat ik in Windows op de PC ingelogd ben als de Windows user NIJP02 sta ik reeds in de lijst met Logins



We kunnen een Active Directory User gebruiken of een nieuwe lokale gebruiker op de PC aanmaken:



## Hoe meldt deze persoon zich aan?

Voer het e-mailadres of telefoonnummer in van de persoon die je wilt toevoegen. Als deze persoon Windows, Office, Outlook.com, OneDrive, Skype of Xbox gebruikt, voer je het e-mailadres of telefoonnummer in dat deze persoon gebruikt om zich aan te melden.

E-mailadres of telefoonnummer

**Ik beschik niet over de aanmeldgegevens van deze persoon**

Annuleren

Volgende



## Account maken

iemand@example.com

Een telefoonnummer gebruiken

Nieuw e-mailadres maken

**Gebruiker zonder Microsoft-account toevoegen**

Terug

Volgende

Microsoft-account

Een gebruiker voor deze pc maken

Als u een wachtwoord wilt gebruiken, kies dan iets dat u gemakkelijk kunt onthouden maar voor anderen moeilijk te raden is.

Wie gaat deze pc gebruiken?

**SQLTest**

Zorg dat het veilig is.

**\*\*\*\*\***

Voor het geval u uw wachtwoord vergeet

Beveiligingsvraag 1

Uw antwoord

Volgende

Vorige

Microsoft-account

Een gebruiker voor deze pc maken

Als u een wachtwoord wilt gebruiken, kies dan iets dat u gemakkelijk kunt onthouden maar voor anderen moeilijk te raden is.

Wie gaat deze pc gebruiken?

**SQLTest**

Zorg dat het veilig is.

**\*\*\*\*\***

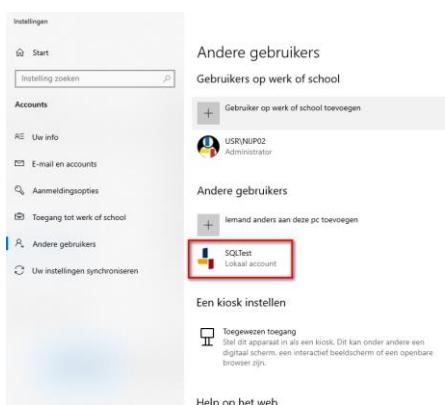
Voor het geval u uw wachtwoord vergeet

Hoe heette uw eerste huisdier?

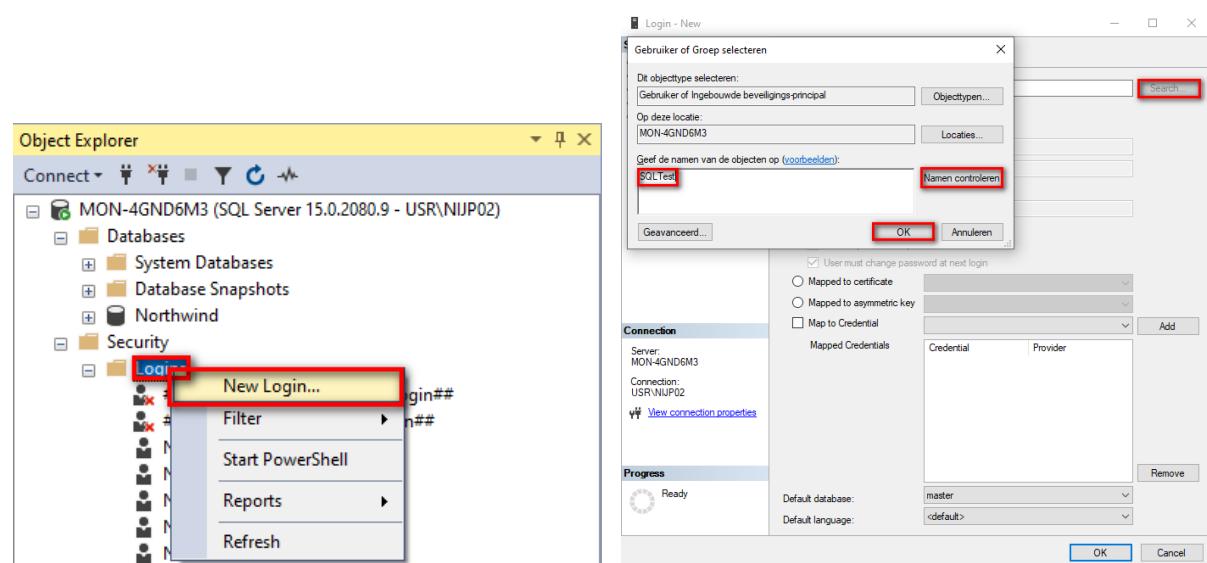
**Goudvis**

Volgende

Vorige

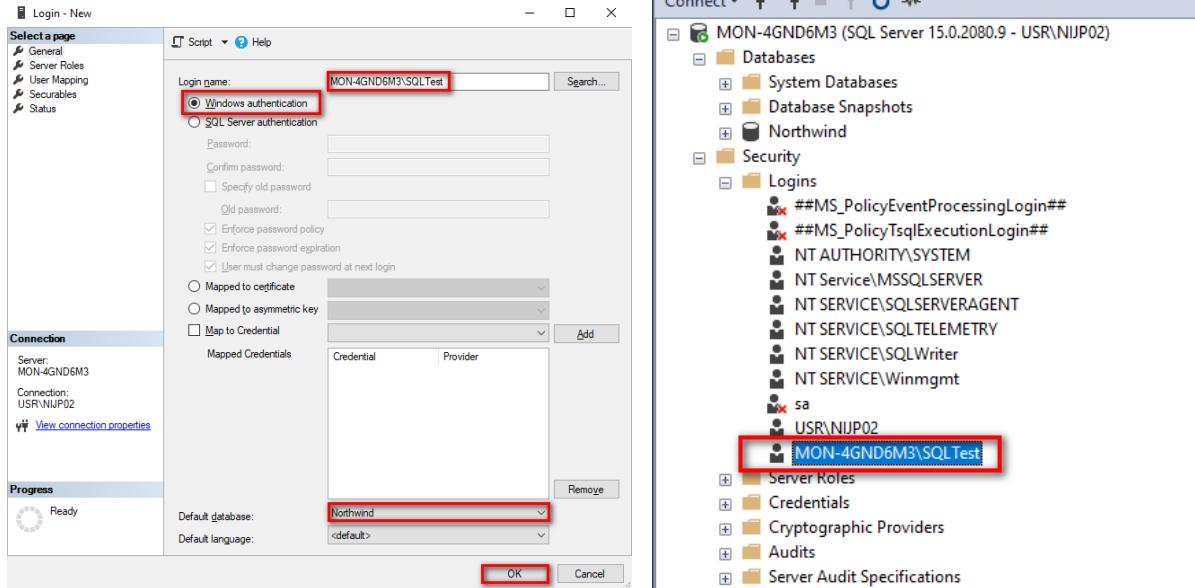


Het Windows account kunnen we vervolgens als Login toevoegen aan SQL Server:



Er moet nu gekozen worden tussen Windows Authentication of SQL Server Authentication. Windows Authentication is ook wel bekend als Integrated Security omdat het SQL Server security model sterk geïntegreerd is met het Windows security model en is de voorkeur van Microsoft. De gebruiker moet dan eerst op Windows inloggen voordat het SQL Server kan gebruiken.

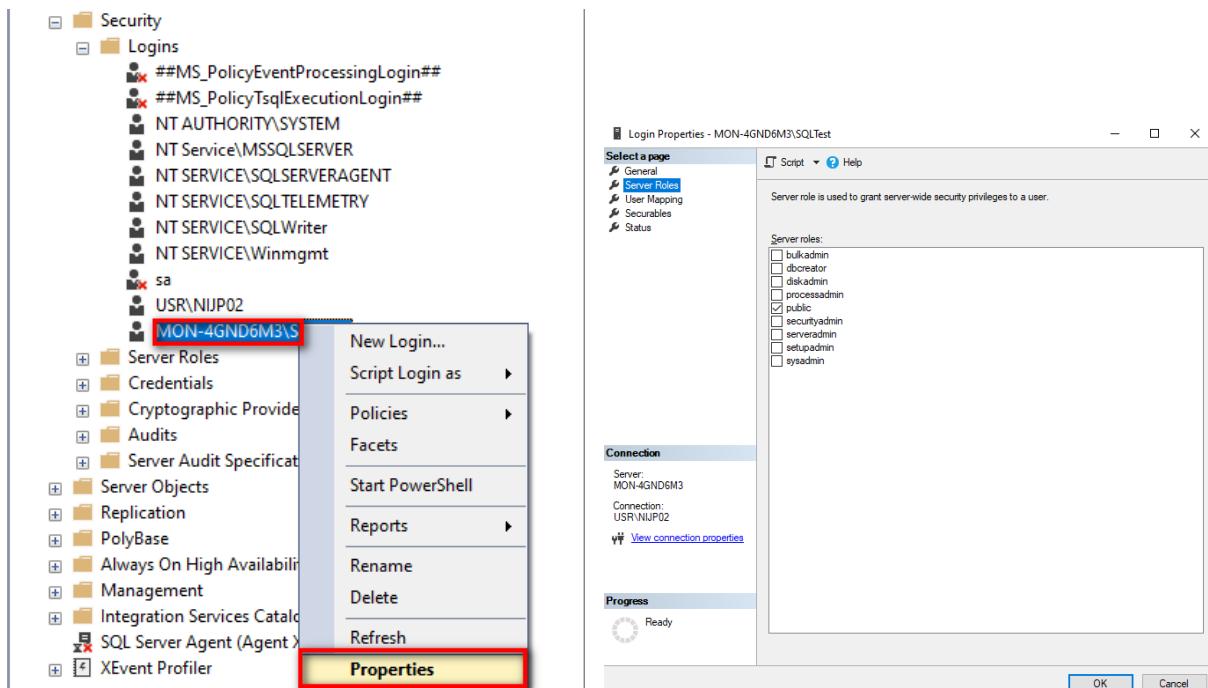
Selecteer vervolgens de Default database waar de Login in het begin naar moet kijken. Je kan elke database selecteren maar dat wil nog niet zeggen dat je die deur binnen mag. Als je geen toegang tot de database hebt dan sta je slechts voor de deur van de database. Het heeft daarom de voorkeur om een database te selecteren waar de gebruiker wel toegang toe heeft of krijgt (in dit geval Northwind). De nieuwe Login SQLTest wordt nu weergegeven in de lijst met Logins.



Met de Login kunnen we nu wel binnen de stadsmuren maar nog niet in een database.

### 23.8 Server Roles

Als we naar de Properties van de Login kijken dan zien we 9 Server Roles die we aan de Login kunnen toekennen:



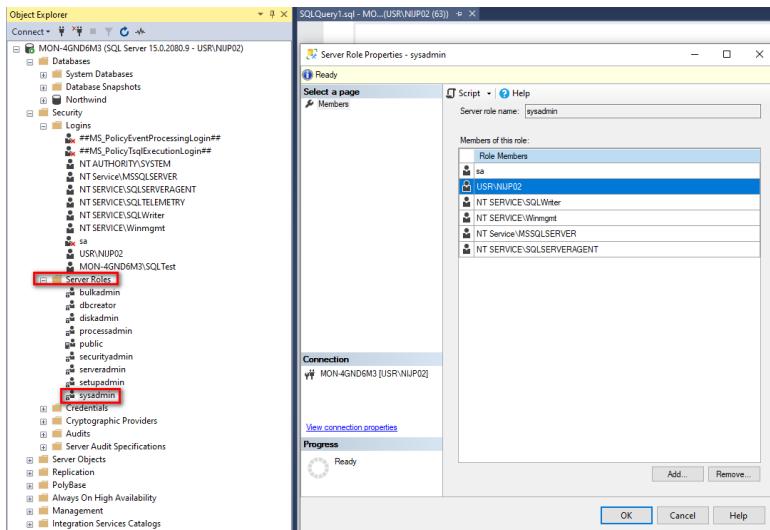
Deze 9 Fixed Server Roles hebben niets te doen met de databases zelf maar met de SQL Server Engine:

sysadmin	Deze meest belangrijke Server Role bevat alle onderstaande Server Roles plus nog meer en kan alle activiteiten op de server uitvoeren
----------	---

serveradmin	Dit laat gebruikers toe om serverbrede configuratie opties uit te voeren, inclusief het down brengen van de server
securityadmin	Dit laat de beveiligingsbeheerder toe om op serverniveau en databaseniveau rechten toe te kennen (Grant) of te weigeren (Deny), inclusief het resetten van paswoorden
processadmin	Dit kan draaiende processen stoppen
setupadmin	Kan gekoppelde servers toevoegen of verwijderen, b.v. wanneer we een server ergens anders op het netwerk hebben en we willen dat de twee server makkelijk met elkaar kunnen communiceren
bulkadmin	Dit is alleen wanneer we gebruik maken van bulk insert statements (Bulk Insert Recovery Model)
diskadmin	Dit is voor het beheren van de bestanden op schijf
dbcreator	Dit is om databases aan te maken en te verwijderen, maar ook voor het restoren van een database
public	Dit is eigenlijk niet een echte Fixed Role omdat je er serverrechten aan kan toevoegen

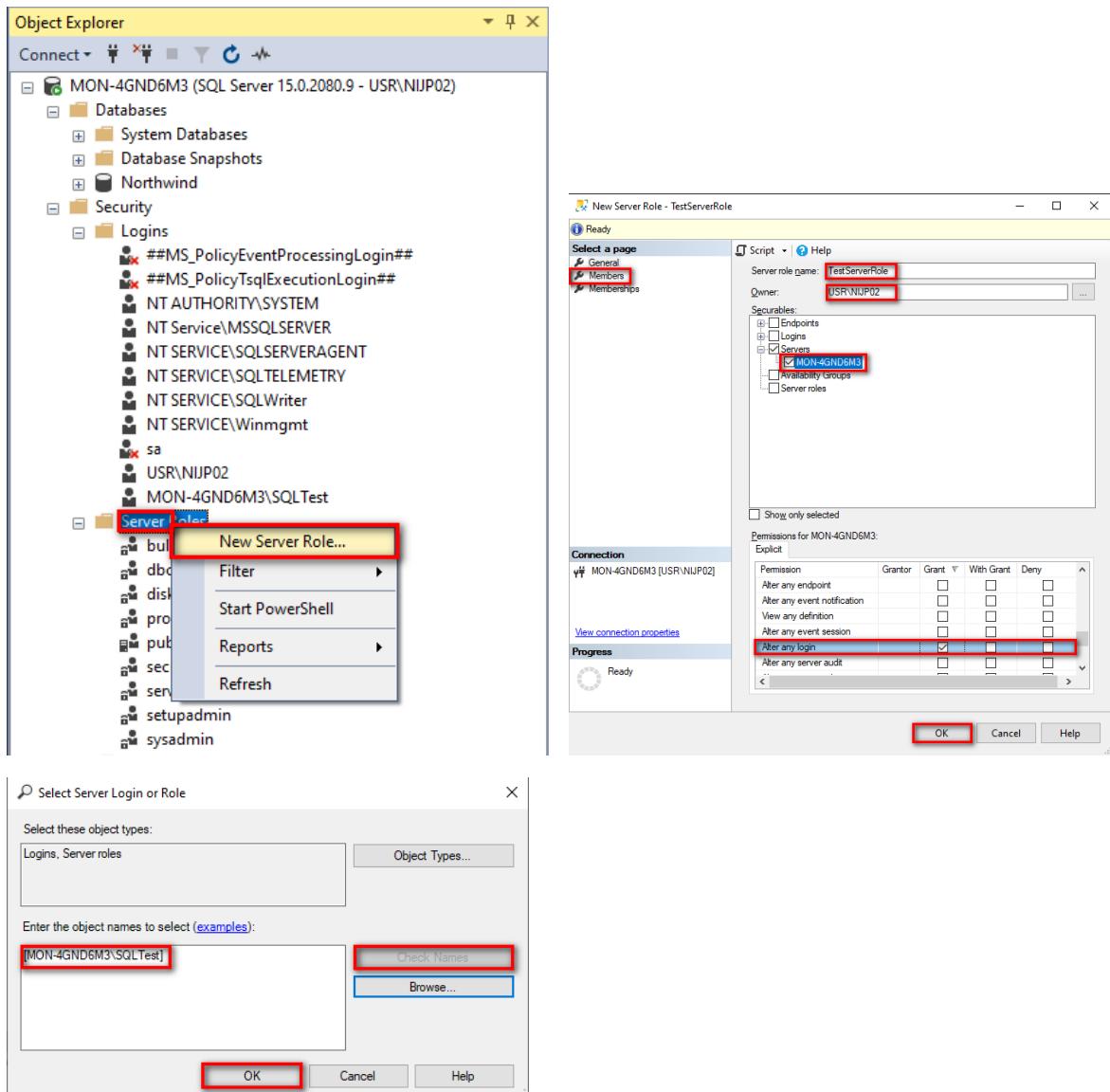
Voor meer uitleg over SQL Server Roles and Permissions, zie <https://docs.microsoft.com/en-us/sql relational-databases/security/authentication-access/server-level-roles?view=sql-server-ver15>

Door naar de properties van de Server Roles te kijken kan je ook zien welke gebruikers die betreffende Role toegekend hebben gekregen:

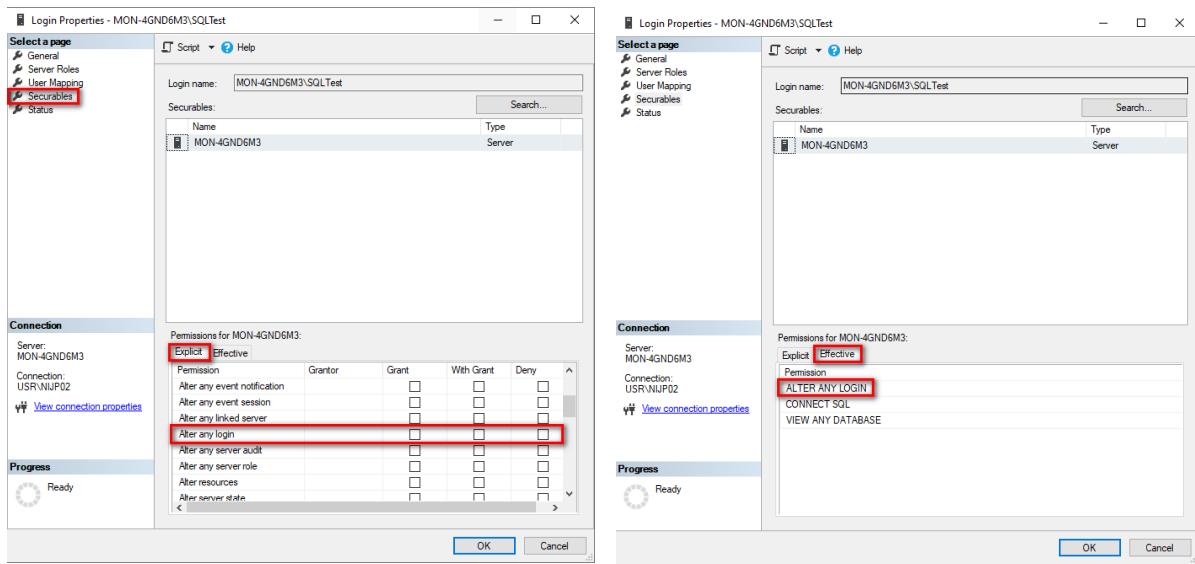


De bedoeling van security is om de gebruikers het minimale aantal privileges toe te kennen die ze nodig hebben.

Naast de Fixed Server Roles kunnen we ook door gebruikers gedefinieerde Server Roles aanmaken door de Server role name, Owner, Securables (=objecten waarop privileges kunnen worden gegeven), de permissies te selecteren en een member van de role toe te kennen.



Wanneer we nu naar de properties van de aangemaakte Login SQLTest kijken dan zien we dat de Login geen expliciete ALTER ANY LOGIN privileges heeft maar wel effectief deze privilege heeft via de aangemaakte role:



Bij het maken van de role konden we drie soorten expliciete permissies toekennen:

- Grant = privilege toekennen
- With Grant = privilege toekennen plus de privilege om de betreffende privilege ook weer te mogen toekennen aan andere Logins
- Deny = weigeren van de privilege

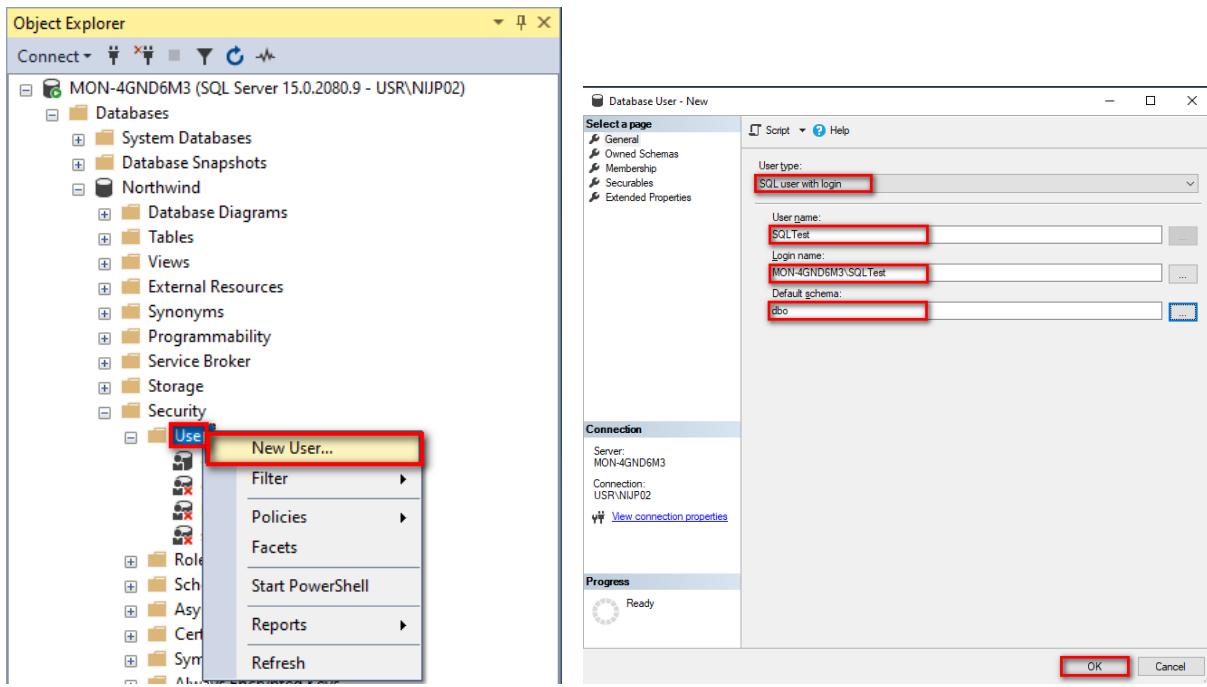
Wanneer er nu twee verschillende Roles zijn toegekend aan een Login waarbij de ene Role een grant toekent maar de andere Role een Deny, dan is het effectieve resultaat dat de privilege niet toegekend is omdat de Deny sterker is dan de Grant.

Met Revoke kunnen we Grants of Deny's toekenningen weer weg nemen.

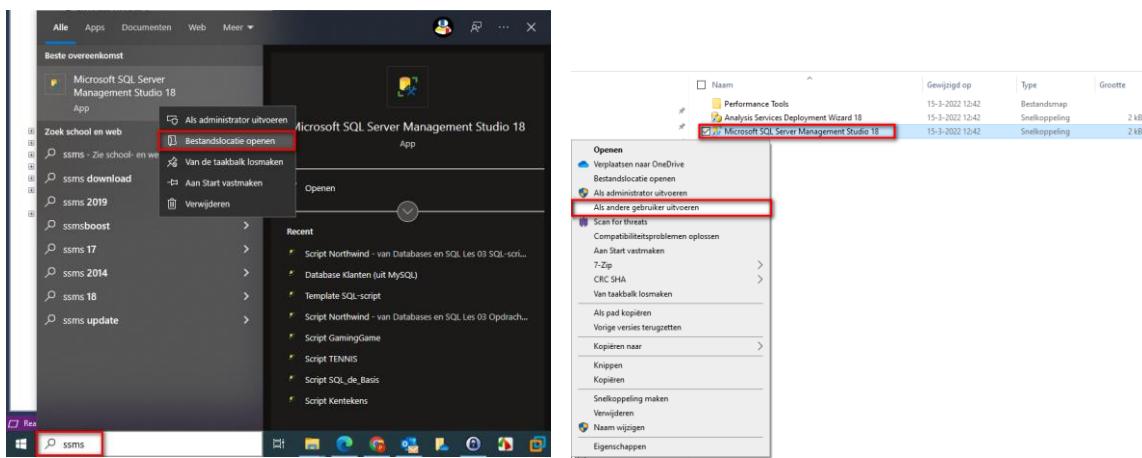
### 23.9 Users

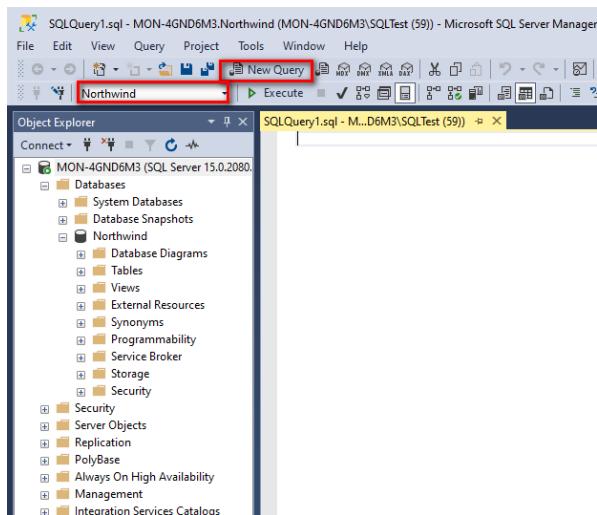
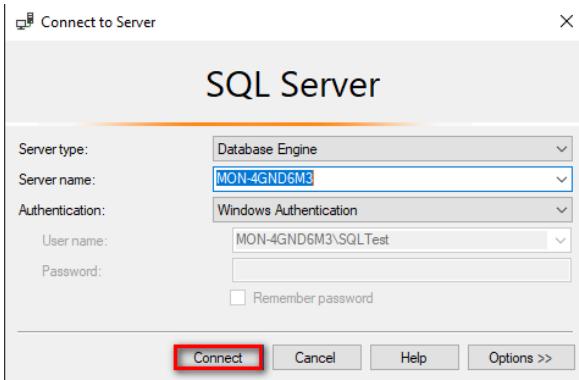
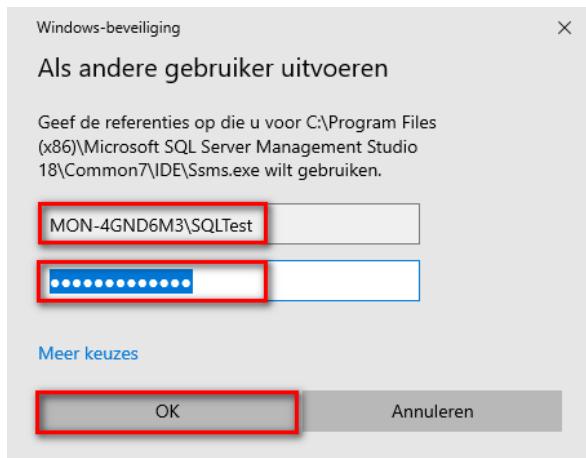
De aangemaakte Login kan nog niet in de database Northwind komen omdat er voor de Login nog geen User is aangemaakt in deze database.

We kunnen een nieuwe User in de database aanmaken door binnen de database (Northwind) met de rechter muisklik op Users -> New User -> SQL user with login -> Username = <bij voorkeur dezelfde naam als de Login, maar mag ook andere naam zijn> -> Login name = <naam van de Login die aan de user gekoppeld moet worden> -> Default schema = dbo <dbo = database owner, maar ander schema mag ook> -> OK

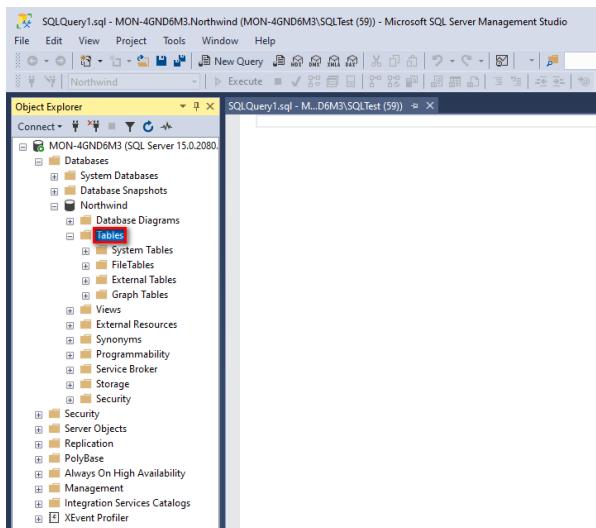


We kunnen nu met de Login SQLTest inloggen in SQL Server en komen terecht in de Northwind database door : zoek naar SSMS -> rechtermuisklik -> Bestandslocatie openen -> Met SHIFT knop ingedrukt rechtermuisknop SSMS uitvoeren als andere gebruiker uitvoeren -> Windows gebruikers naam = SQLTest -> paswoord -> OK -> Connect -> New Query -> controleer dat je in de Northwind database zit



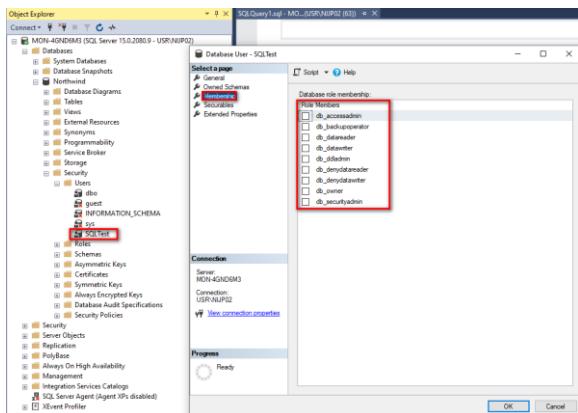


Wanneer de User SQLTest echter binnen de database Northwind kijkt naar de tabellen dan ziet hij echter géén tabellen. We zijn met onze Login SQLTest in SQL Server binnen gekomen en met onze User SQLTest de database binnen gekomen maar we kunnen nog niets zien in de database. Ter vergelijking zijn we met onze Login de stad binnengekomen en met onze User de woning binnen gekomen maar staan nu in de gang van de woning maar hebben geen toegang tot de kamers in de woning.



## 23.10 Database Roles

Om de User iets te laten zien in de database moeten we de User Roles of Permissies toekennen binnen de database. Net zoals we Fixed Server Roles hebben, hebben we ook 9 Fixed Database Roles en onze nieuwe User SQLTest heeft nog geen van de rollen toegekend gekregen:



<b>db_owner</b>	Dit laat alles toe binnen de database (net zoals de sysadmin Server Role), bijvoorbeeld configuratie en onderhoud activiteiten en database verwijderen uit SQL Server
<b>db_securityadmin</b>	Dit kan rollidmaatschap (role membership) modifiteren en permissies beheren
<b>db_accessadmin</b>	Dit laat een specifieke gebruiker toe om toegang tot de database toe te voegen of te verwijderen voor Windows Logins, Windows Groups, en SQL Server Logins m.b.v. CONNECT en ALTER ANY USER
<b>db_backupoperator</b>	Dit laat de User toe om backups te maken van de database
<b>db_ddladmin</b>	Dit laat de User toe om alle DDL commando's te gebruiken
<b>db_datawriter</b>	Dit laat de User toe om data toe te voegen (ADD), te verwijderen (DELETE), te veranderen (UPDATE) in alle user tables
<b>db_denydatawriter</b>	Dit voorkomt dat de User data kan toevoegen (ADD), verwijderen (DELETE), te veranderen (UPDATE) in alle user tables. DENY is sterker dan GRANT!
<b>db_datareader</b>	Dit laat de User toe om de data in de user tables te lezen (READ)

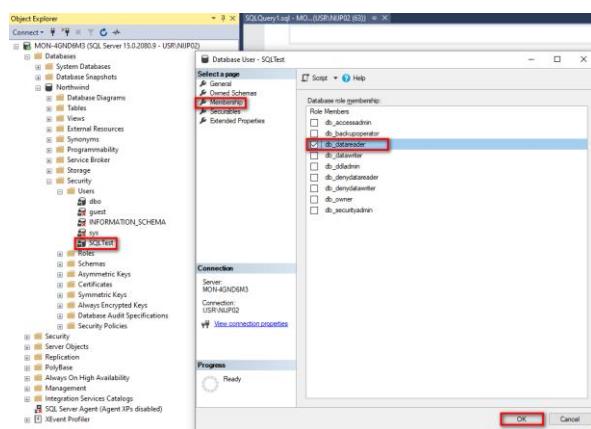
db_denydatareader	Dit voorkomt dat de User data in de user tables kan lezen (READ). DENY is sterker dan GRANT!
public	Default zijn er geen database level permissions aan public toegekend

Let op: Daar DENY sterker is dan GRANT kunnen we dus een User toestaan om data te manipuleren maar niet terug kan lezen of andersom

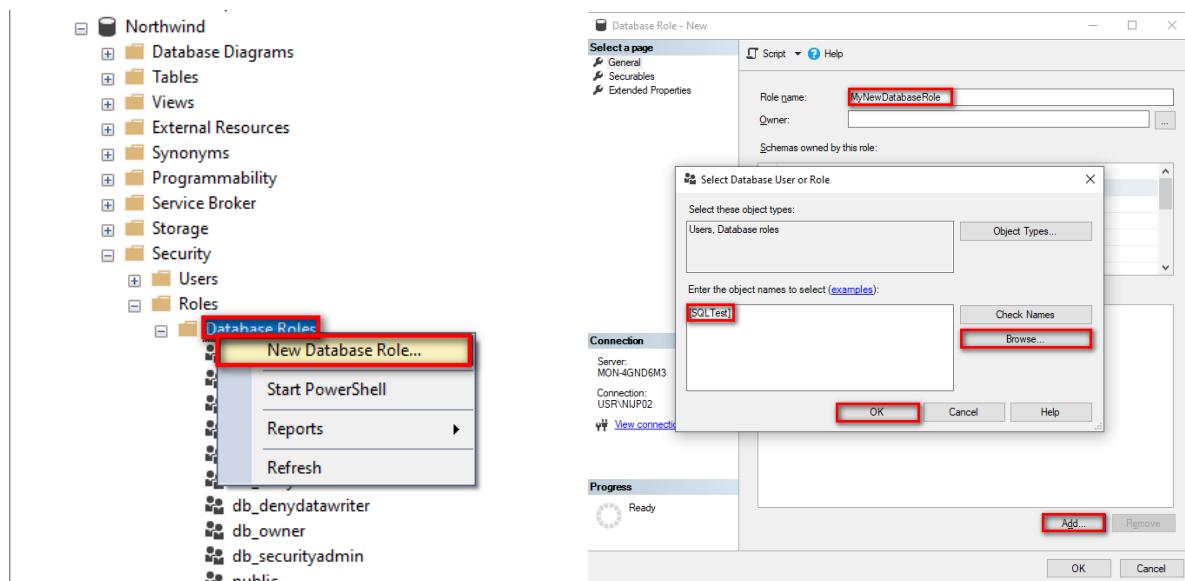
Voor meer uitleg over SQL Server Database Roles and Permissions, zie

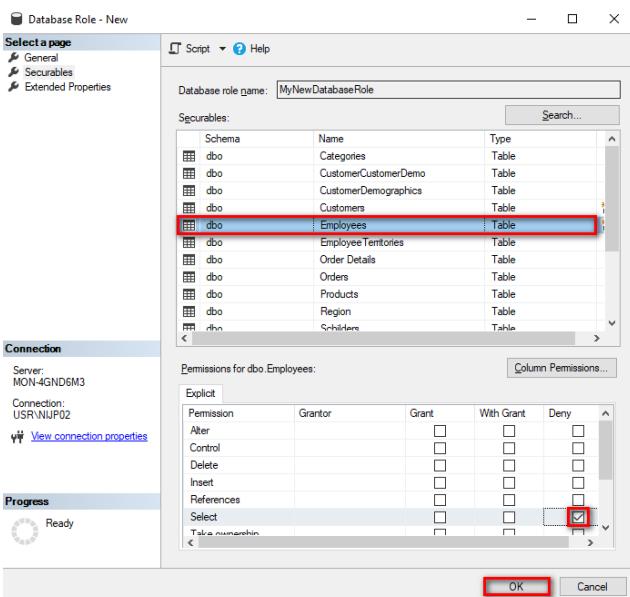
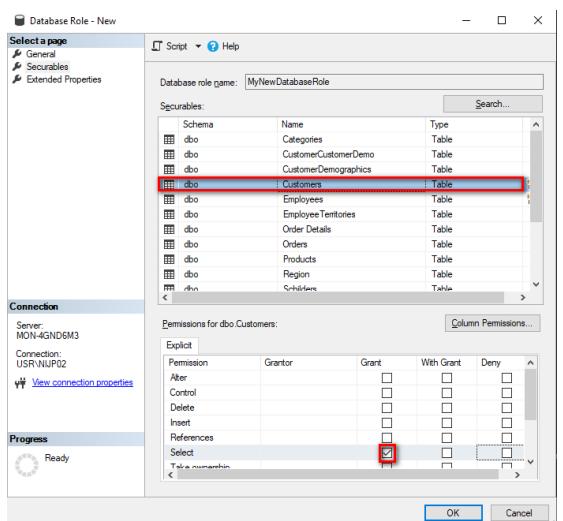
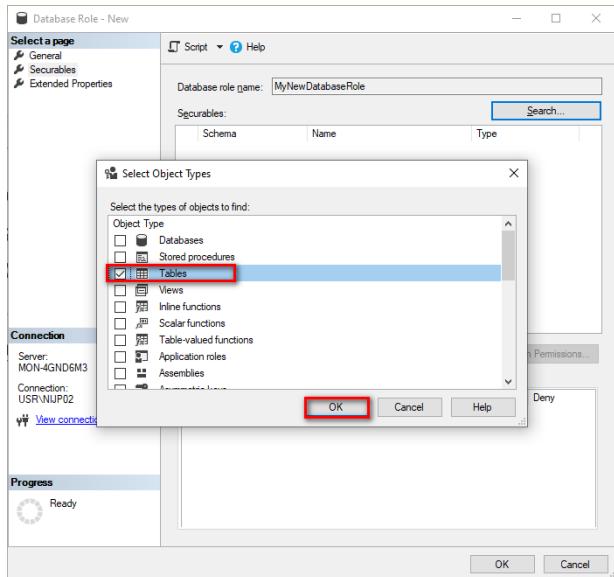
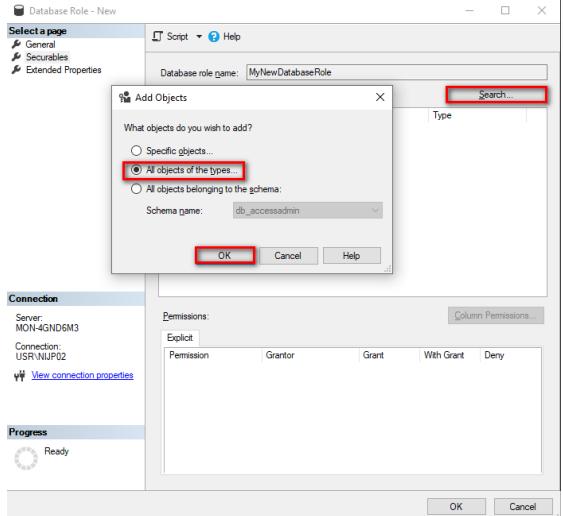
<https://docs.microsoft.com/en-us/sql/relational-databases/security/authentication-access/database-level-roles?view=sql-server-ver15>

We kunnen een specifieke User toekennen aan de Database Role door rechtermuisklik op de User -> Properties -> Membership -> Selecteer de Database Role die je aan de user wilt toekennen

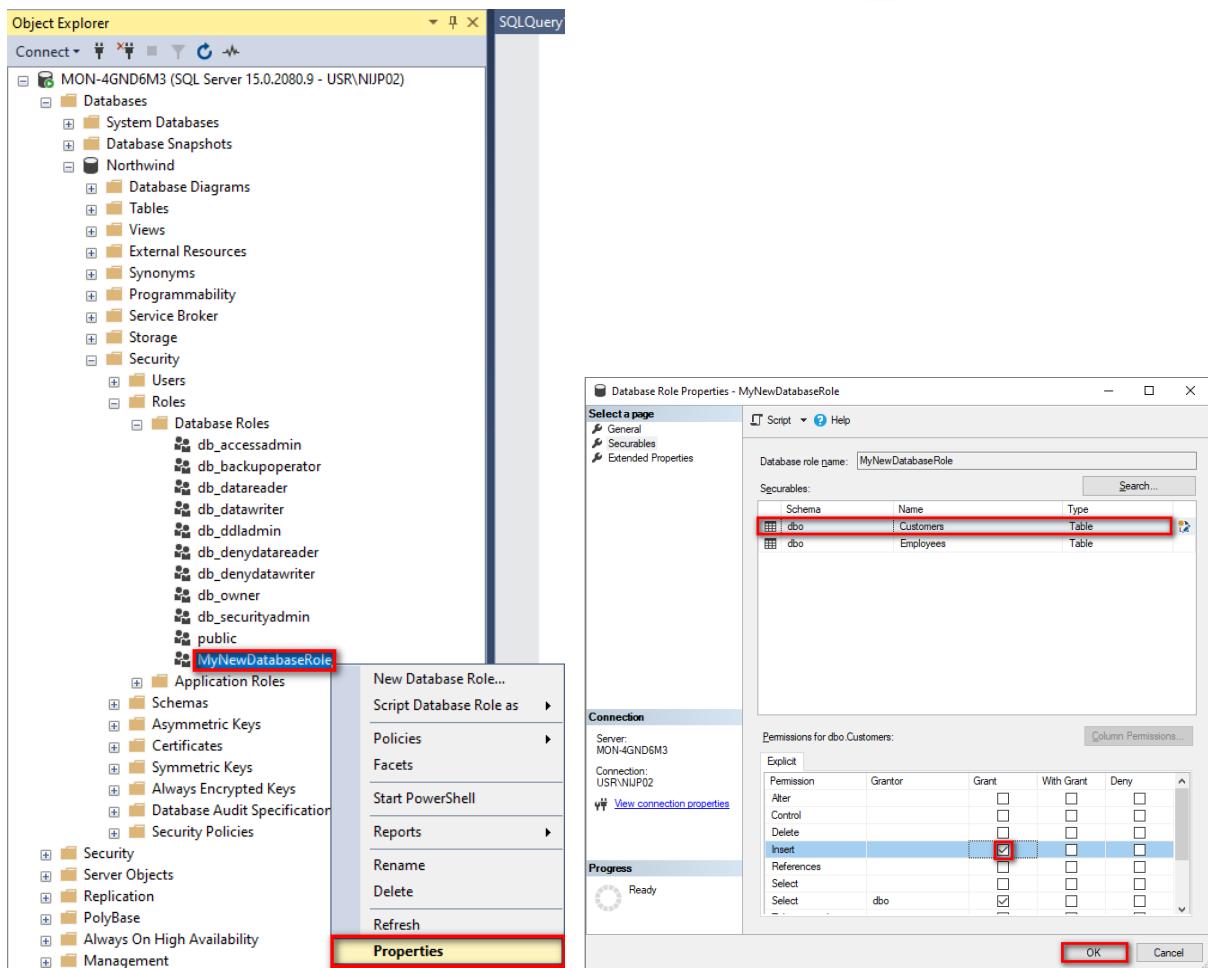


Net zoals we nieuwe Server Roles kunnen aanmaken kunnen we ook nieuwe database Roles aanmaken en toekennen aan net zoveel gebruikers als je wilt met rechtermuisklik op Database Roles van de database -> New Database Role -> Role name = <sprekende naam> -> Add -> Browse -> Selecteer Users en/of Database Roles die Role toegekend moeten krijgen -> OK -> Securables -> Search -> All objects of the types -> OK -> Tables -> OK -> Selecteer de expliciete permissies per securable (in dit geval Grant Select op tabel Customers en Deny Select op tabel Employees) -> OK





De aangemaakte Database Role kunnen we vervolgens aanpassen door Rechtermuisklik op de betreffende Database Role -> Properties -> Securables -> Selecteer expliciete permissies (in dit geval Grant Insert op tabel Customers) -> OK



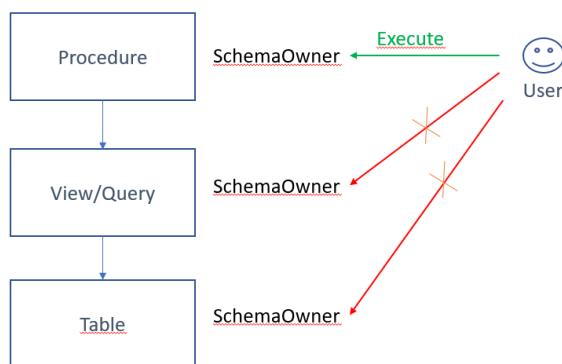
De expliciete permissies die we kunnen toevoegen zijn o.a.:

Alter	Hiermee kan je de permissies van een securable veranderen, behalve de ownership. Dus hiermee kan ALTER, CREATE, DROP op elke securable
Control	Hiermee kan je de ownership van de securable aanpassen, dus heb je al de permissies op de securable
Delete	Hiermee kunnen rijen verwijderd worden uit een securable met DELETE
Insert	Hiermee kunnen rijen toevoegen worden in een securable met INSERT
References	Hiermee kunnen Foreign Key constraints worden gemaakt naar de securable
Select	Hiermee kunnen gegevens uit de securable worden opgehaald
Update	Hiermee kunnen gegevens in de securable worden aangepast
Execute	Hiermee kan een Stored Procedure opgestart worden
Take ownership	
View change tracking	Hiermee kan worden gezien welke Roles zijn gewijzigd
View definition	Hiermee kan naar de metadata van de securable worden gekeken

Het is mogelijk om voor een Database Role specifieke securables of alle securable, specifieke schemas of alle schemas expliciete permissies te geven. Roles kunnen genest worden.

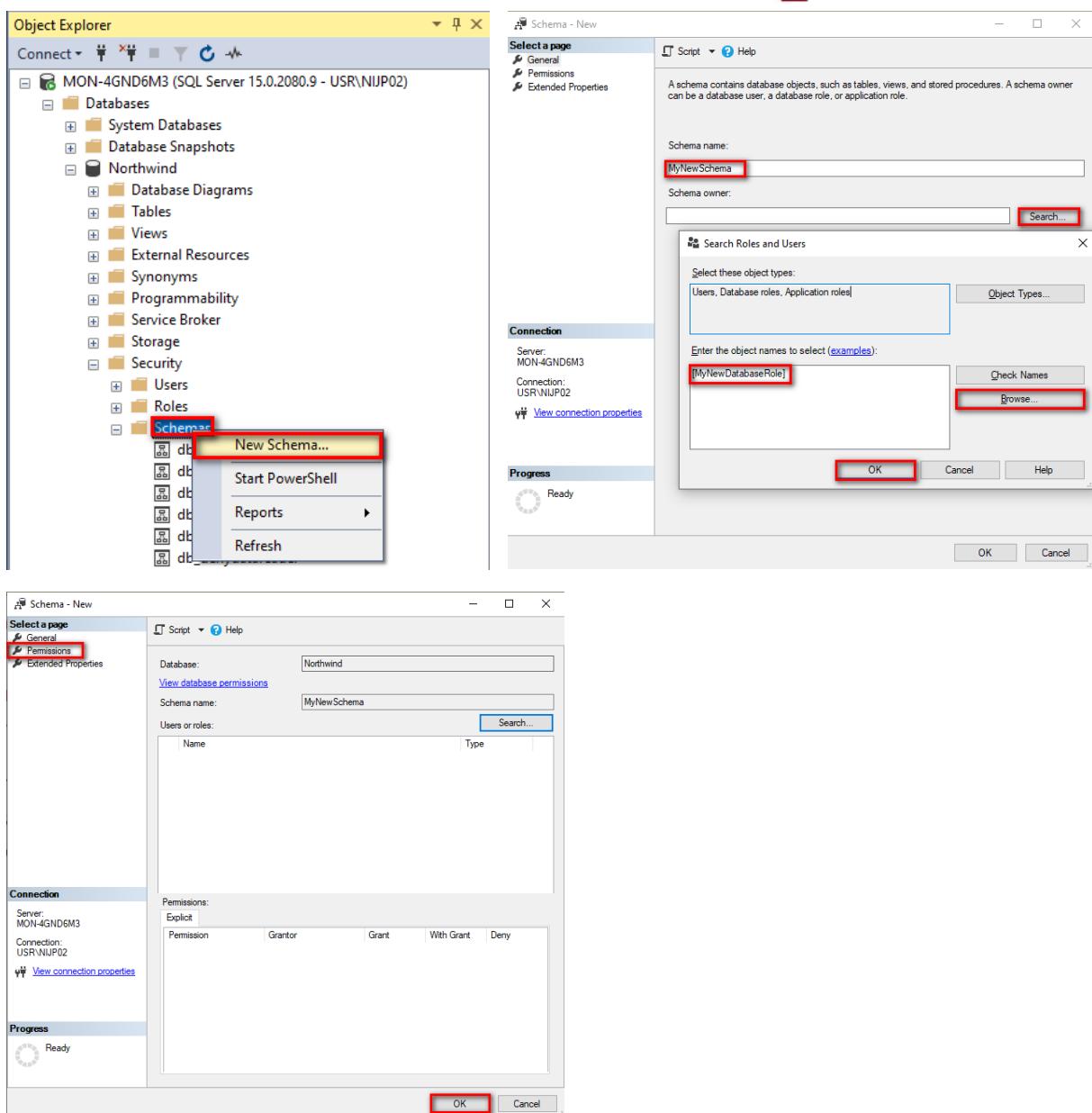
### 23.11 Schemas

Een schema is het eerste deel in de naam van een tabel zoals HumanResources in HumanResources.Department. De tabel Department maakt dus deel uit van het Schema HumanResources. We kunnen een Owner hebben voor een specifiek schema. De Owner van een schema is belangrijk i.v.m. Ownership Chaining. b.v. Een Procedure is gebouwd op een View, welke een stored SQL-query is. De procedure is afhankelijk van de onderliggende view en de view is afhankelijk van de onderliggende tabel. Dit is een voorbeeld van Ownership Chaining, de procedure gaat naar view en die gaat naar tabel. De user heeft Execute permissions op de Stored Procedure maar géén permissions op de onderliggende view en tabel. Zolang de Stored Procedure dezelfde SchemaOwner heeft als de onderliggende view en tabel dan controleert SQL Server niet of de User privileges heeft op deze view en tabel. Doordat de User alleen Execute rechten nodig heeft op de procedure kan de User dus alleen de view en tabel benaderen via deze procedure, en dat komt de security ten goede. Wanneer echter de tabel een andere SchemaOwner heeft en de User de procedure aanroept dan controleert SQL Server niet of de User permissions heeft op de view maar wel of de User permissions heeft op de tabel.



Met Schema's kunnen we dus gerelateerde objecten groeperen. Wanneer we één ervan benaderen dan benaderen we ze allen via die ene. Het is niet noodzakelijk dat de Schema's hetzelfde zijn maar de SchemaOwners moeten hiervoor hetzelfde zijn. De Schema's mogen dus van elkaar verschillen, zolang de SchemaOwners maar dezelfde zijn.

We kunnen nieuwe schema's binnen een database aanmaken door rechtermuisklik op Schemas -> New schema -> Schema name = (voer sprekende naam voor schema in) -> Search -> Browse -> (selecteer User of Database Role als Owner voor het schema) -> OK -> Permissions -> voeg indien gewenst de gewenste Roles en Permissions toe -> OK



## Hoofdstuk 24 CAPACITY MANAGEMENT

### 24.1 Capacity Management

Nadat de database beheerder ervoor heeft gezorgd dat er nooit data verloren zal gaan (Backup & Recovery) en dat elke gebruiker slechts bij de gegevens kan komen waartoe hij voor zijn functie toegang moet hebben (User Administration) is het de taak voor de databasebeheerder om ervoor te zorgen dat de database niet onverwachts volledig vol kan raken, maar dat de capaciteit beantwoordt aan de huidige en toekomstige behoeften op kosteneffectieve wijze (Capacity Management).

Een database management system bestaat uit een aantal processen, een stuk gealloceerd geheugen en database bestanden. De processen zijn uitvoerbare bestanden (executables) die processorcapaciteit in gebruik nemen. Deze processen hebben een stuk gealloceerd geheugen nodig om gegevens in te kunnen verwerken. De te verwerken gegevens komen uit de databasetabellen die opgeslagen zijn in de database bestanden.

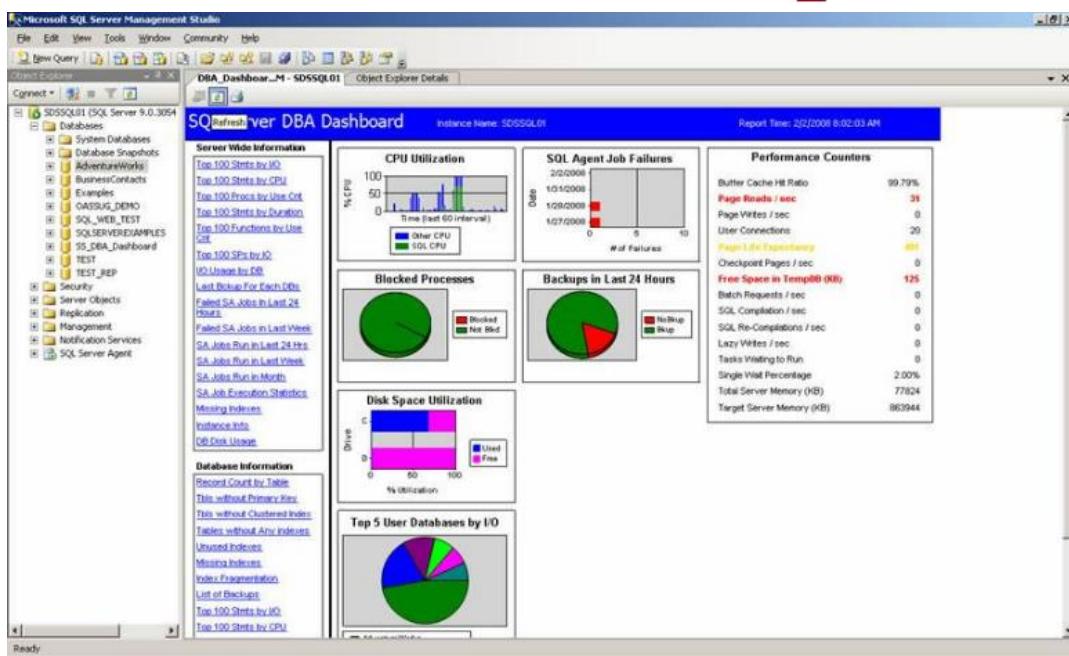
Deze database bestanden staan op filesystems of logische schijven (Virtual Hard Disks), welke weer via SAN (Storage Area Network) of RAID systemen op fysieke harde schijven staan. Elk filesystem, logische schijf of fysieke harde schijf is beperkt in grootte en heeft dus een maximale grootte.

In databases wordt dagelijks gegevens toegevoegd en gewijzigd, waardoor de tabellen groeien totdat het tegen een limiet aanloopt waarbij één van de databasebestanden volledig vol is geraakt. De database komt dan in een bevroren toestand te staan omdat het geen gegevens meer in de database kan toevoegen of wijzigen. Als gevolg hiervan kunnen de gebruikers dan geen gebruik meer maken van de database totdat het probleem is opgelost. Vaak is het slechts een kwestie van uitbreiding van disks, maar soms dient er een volledige nieuwe configuratie te worden uitgevoerd.

Zolang de bevroren toestand op de database aanwezig is kan er géén gebruik worden gemaakt van de database. Er kunnen zelfs situaties ontstaan waardoor zelfs de databasebeheerder door de bevroren situatie niet meer kan inloggen op de database. Zolang de database in een bevroren toestand staat kan het bedrijf haar primaire proces niet meer uitvoeren en gaat de situatie het bedrijf uiteindelijk geld kosten. Een bevroren toestand van de database is dus een ongewenste situatie die de databasebeheerder te allen tijde moet weten te voorkomen.

Een uitbreiding van diskruimte kan dagen of weken duren terwijl het voor het bedrijf niet acceptabel is dat de database gedurende die tijd niet beschikbaar is. Het is daarom voor de database beheerder van groot belang dat hij te allen tijde weet hoeveel vrije ruimte er nog beschikbaar is zodat hij/zij tijdig een plan kan maken voor een uitbreiding en er nog tijd is om de schijven te bestellen en te configureren.

Capacity Management hoeft niet moeilijk te zijn maar het is wel belangrijk dat er continu op de beschikbare capaciteit gemonitord wordt. Er bestaan verschillende monitoring tools die continu de status informatie van de operation systems en de databases verzamelt en weergeeft in een dashboard scherm, zoals in het onderstaande voorbeeld:



Vaak zijn deze tools in staat om alerts te sturen naar het Incident Management System om geautomatiseerd een incident aan te maken. De database beheerders kunnen dan op basis van de incidenten in het Incident Management System de problemen oplossen. Bij dingende problemen kan vanuit het Incident Management System direct contact worden gezocht met de verantwoordelijke database beheerder (bij 24x7 systemen loopt altijd één van de database beheerders een standby dienst en dient dan dag en nacht bereikbaar te zijn om direct problemen op te lossen).

Indien er geen monitoring tool aanwezig is (ze zijn redelijk duur), dan dient de databasebeheerder zelf (SQL) scripts te maken om de status van de database periodiek te monitoren.

Een voorbeeld SQL query om een lijst van de bestanden van de aanwezige databases te verkrijgen is:

```
SELECT name 'Logical Name', physical_name 'File Location'
FROM sys.master_files;
```

Een voorbeeld SQL query om informatie over de vrije ruimte in de database bestanden van de database Northwind te verkrijgen is:

```
USE Northwind;
SELECT DB_NAME() AS DbName, name AS FileName, type_desc, size/128.0 AS CurrentSizeMB,
size/128.0 - CAST(FILEPROPERTY(name, 'SpaceUsed') AS INT)/128.0 AS FreeSpaceMB
FROM sys.database_files;
```

Een truuk die in de praktijk wel toegepast wordt om ad-hoc problemen te kunnen oplossen is het maken van een dummy bestand met een grootte van ca. 10% van de schijfgrootte op elk filesystem of logische schijf. In geval dat een filesystem of logische schijf vol is geraakt kan de databasebeheerder het ad-hoc probleem oplossen door het dummy bestand te verwijderen, waardoor er weer 10% ruimte vrij komt op de schijf en de database beheerder alsnog de nodige schijfuitbreidingsplannen kan organiseren.

Een voorbeeld commando om een dummy bestand van 10 GB aan te maken is:

Linux	: <b>fallocate -l 10G dummy</b>
Windows	: <b>fsutil file createnew dummy 10737418240</b>

## Hoofdstuk 25 PERFORMANCE TUNING - **TODO**

Nadat de database beheerder ervoor heeft gezorgd dat er nooit data verloren zal gaan (Backup & Recovery), dat elke gebruiker slechts bij de gegevens kan komen waartoe hij voor zijn functie toegang moet hebben (User Administration) en ervoor heeft gezorgd dat de database niet onverwachts volledig vol kan raken (Capacity Management) is het taak voor de database beheerder om ervoor te zorgen dat de database continu een goede performance (responsetijd) blijft behouden.

Performance problemen kunnen door verschillende redenen worden veroorzaakt. Soms zijn performance problemen langzaam groter geworden met de groei van de database, maar het kan ook voorkomen dat de performance plotseling erg slecht is geworden.

Bij performance problemen is het de taak van de databasebeheerder om te analyseren wat de oorzaak van de performance problemen zijn, en deze performance problemen zo snel mogelijk op te lossen.

Verschillende oorzaken van database performance problemen kunnen zijn:

- Er zijn problemen op netwerkniveau  
Wanneer er problemen zijn op het netwerk, dan zullen alle of meerdere gebruikers daar last van ervaren en is het aannemelijk dat de database performance problemen ook te maken hebben met de netwerkproblemen. Immers wanneer het netwerk erg traag is dan zal ook de communicatie tussen de clients (PC's) en de database traag zijn. Het heeft dan nog niet veel zin om een database performance analyse op te starten daar de kans er dan in zit dat je werk doet die achteraf onnodig blijkt te zijn geweest.  
In dit geval dient de databasebeheerder contact op te nemen met de netwerkbeheerder (als hij dat niet ook zelf is) en dient er op netwerk niveau een analyse te worden uitgevoerd.  
Soms blijken de netwerkproblemen een gevolg van wijzigingen (change) door de afdeling netwerkbeheer, of soms betreft het een storing in een netwerk device (b.v. netwerk switch).
- Er zijn problemen op serverniveau  
Wanneer er performance problemen zijn op de server waar de database op draait, dan kan het zijn dat het RDBMS de veroorzaker is van de performance problemen, maar het kan ook zijn dat de performanceproblemen een gevolg zijn van andere problemen op de server.  
Immers als de server traag is dan zal het RDBMS ook traag worden. Het heeft dan nog niet veel zin om een database performance analyse op te starten daar de kans er dan in zit dat je werk doet die achteraf onnodig blijkt te zijn geweest.  
In dit geval dient de databasebeheerder contact op te nemen met de serverbeheerder (als hij dat niet ook zelf is) en dient er op server niveau een analyse te worden uitgevoerd.  
Soms blijken de serverproblemen een gevolg van wijzigingen (change) door de afdeling serverbeheer, of soms betreft het een storing in een Server component (b.v. netwerkkaart).
- Er zijn problemen op databaseniveau  
In dit geval dient de databasebeheerder zelf een analyse op de database uit te voeren. Zo nodig kan de databasebeheerder een root cause analyse uitvoeren samen met de applicatiebeheerder, serverbeheerder en netwerkbeheerder.

In dit hoofdstuk zullen wij ons richten op problemen op databaseniveau.

DDL en DCL instructies hebben veelal een constante verwerkingstijd, maar de verwerkingstijd van DML instructies zijn erg afhankelijk van de gegevens in de tabellen. Drie technieken om de responsetijd van DML instructies te versnellen zijn denormaliseren, indexeren en het anders formuleren van instructies. Deze drie technieken zullen in de volgende drie paragrafen worden besproken.

## 25.1 Denormaliseren

Door het normaliseren tijdens het ontwerp van de database (zie paragraaf 1.2) zijn de gegevens per gegevensgroep in aparte tabellen opgeslagen en is voorkomen dat er redundante gegevens in de database komen te staan.

Om gewenste informatie uit de database te halen moet (door de normalisatie) gegevens uit diverse tabellen worden samengevoegd m.b.v. JOINS. JOINS van meerdere (grote) tabellen kunnen bij het uitvoeren van SELECT instructies erg tijdrovend zijn. Een manier om SELECT instructies te versnellen is om dan toch maar (tegen het normaliseren in) redundante gegevens op te nemen in de database. Dit toevoegen van redundante gegevens wordt ook wel **denormaliseren** genoemd.

Door het denormaliseren kunnen SELECT instructies sneller worden, echter zullen UPDATE en DELETE instructies daardoor juist weer trager worden. Een ander nadeel van denormalisatie is dat de dubbel opgeslagen gegevens ook dubbele opslagruimte nodig heeft en er kans ontstaat op inconsistentie binnen de database.

We onderscheiden twee soorten databases, te weten **Online Transaction Processing OLTP** databases en **Online Analytical Processing OLAP** databases. OLTP databases zijn databases waar het primaire proces van het bedrijf mee werkt. In deze databases wordt gedurende de dag intensief gelijktijdig door meerdere gebruikers SELECT, UPDATE en DELETE instructies uitgevoerd. OLAP databases zijn datawarehouses, grote databases met veel historische gegevens. Datawarehouses zijn ontworpen om data analyses (**Business Intelligent BI**) mogelijk te maken. Dagelijks wordt er 's nachts actuele data uit de OLTP databases overgezet naar de datawarehouses zodat de data analyses ook redelijk actuele gegevens bevat voor de berekeningen.

Als voorbeeld kunnen we kijken naar een database van een bank waarin alle actuele tegoeden en schulden van de klanten in staan. Het is belangrijk dat de gegevens in deze database actueel zijn en de responsetijd van de database snel is. Wanneer een klant geld heeft opgehaald en vlak daarna bij een supermarkt boodschappen doet en wil betalen door te pinnen, dan wil de bank wel dat er gekeken wordt naar de actuele stand op de bankrekening om te controleren of er voldoende saldo op de bankrekening staat. De SELECT bij het pinnen in de supermarkt moet immers de laatste stand controleren na de UPDATE van het ophalen van het geld en niet de stand voorafgaand aan het ophalen van het geld. Deze database betreft een OLTP database. Het resultaat van de UPDATE van het geld ophalen is direct zichtbaar bij het pinnen in de supermarkt.

De gegevens uit de OLTP database van de bank wordt 's nachts gekopieerd naar het datawarehouse van de bank. In het datawarehouse voeren data analisten hun berekeningen uit op de gegevens die minder actueel zijn dan in de OLTP database. Data analisten berekenen bijvoorbeeld uit hoeveel pintransacties er per periode zijn uitgevoerd of hoeveel geld mensen op een bepaalde dag hebben uitgegeven in een bepaalde regio. Dit datawarehouse betreft een OLAP database. De OLAP database is groot met veel historische gegevens, minder actueel dan de OLTP database en bevat redundante gegevens. Op het datawarehouse wordt vrijwel uitsluitend SELECT instructies uitgevoerd.

Het denormaliseren (redundante gegevens toevoegen) wordt wel toegepast bij datawarehouses maar niet bij Online Transaction Processing OLTP databases.

## 25.2 Indexen

Wanneer wij een INSERT instructie uitvoeren dan worden de rijen in tabellen opgeslagen in bestanden (**files**). Elk bestand bestaat uit pagina's (**pages**). Elke pagina heeft ruimte voor een aantal rijen en de rijen zijn verspreid over de pagina's.

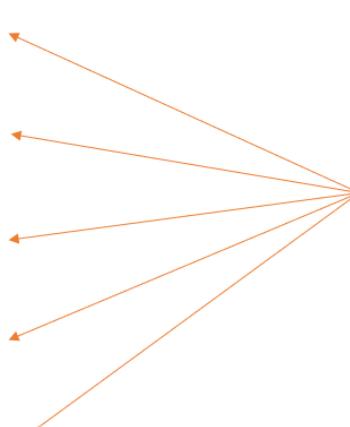
Wanneer er nieuwe rijen worden toegevoegd dan worden ze achter de laatste rij in de laatste pagina opgeslagen en wanneer een pagina vol is dan wordt een lege pagina aan het bestand toegevoegd. Hierdoor zijn alle rijen aaneengesloten. Wanneer er een rij wordt verwijderd dan ontstaat er een gat (ook wel **ghost record** genoemd). Deze gaten worden door SQL niet automatisch gevuld omdat het zoeken naar lege gaten tijdens de schrijfactie te veel tijd kost.

In onderstaande figuur zijn vijf pages van een tabel in een databestand weergegeven waarbij elke pagina vier rijen kan bevatten en er gaten aanwezig zijn.

# Rijen opgeslagen in pagina's

6	Jacobs	...
44	Verhoeven	...
83	De Vries	...
		...
2	Rijks	...
27	Bloem	...
		...
104	Molenaar	...
7	Hogeboom	...
57	De Wit	...
		...
		...
39	Bruinsma	...
112	Guit	...
		...
8	Koops	...
100	Harmsma	...
28	Wigleven	...
		...
95	Gabriel	...

5 Pagina's met elk ruimte voor 4 rijen



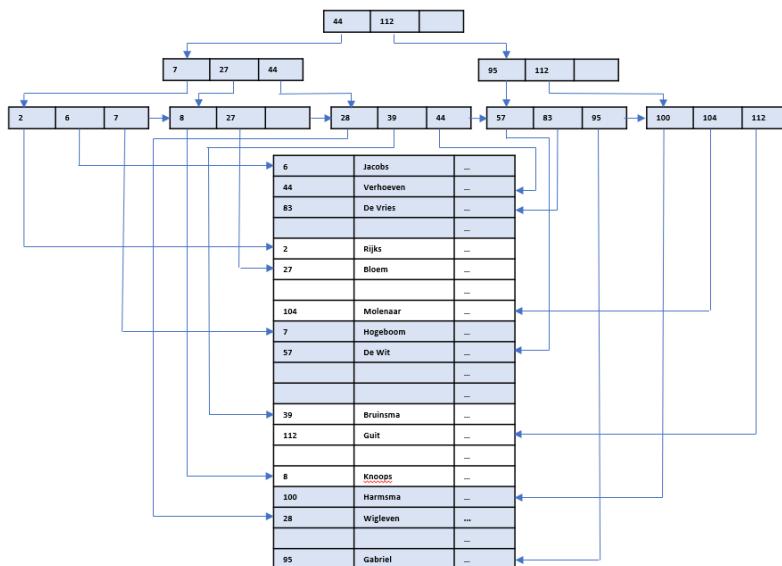
Wanneer het operating system gegevens moet ophalen of schrijven van de harde schijf, dan doet hij dat niet per rij of per byte, maar in pagina's. Gangbare groottes van pagina's zijn 2kB, 4kB, 8kB en 32 kB. Het ophalen van een rij gebeurt in twee stappen. In de eerste stap wordt de pagina van de harde schijf gelezen en in de tweede stap wordt de rij uit de pagina gehaald.

SQL kan op twee manieren de rijen in een tabel benaderen. De eerste methode is de sequentiële zoekmethode waarbij de tabel rij voor rij wordt doorlopen totdat de gewenste rij is gevonden (**Full Table Scan**). Dit kan erg tijdrovend en inefficiënt zijn.

De tweede methode is de geïndexeerde methode waarbij de page en de rij in de tabel via een index wordt benaderd, net zoals we in een boek via de index direct naar de juiste pagina kunnen gaan en daar naar het gezochte woord kunnen zoeken.

Er bestaan verschillende soorten indexen. Een **B-tree index** is een Balanced tree index, waarbij alle takken van de boom ongeveer even lang zijn.

## Zoeken in B-tree indexboom



1. Maak de root het actieve knooppunt
2. Is actieve knooppunt een leafpage?  
Zoja -> ga naar stap 4  
Zonee -> ga naar stap 3
3. Bevat knooppunt gezochte waarde?  
Zoja -> knooppunt wordt actieve knooppunt -> ga terug naar stap 2  
Zonee -> bepaal de kleinste waarde in knooppunt die groter is dan de gezochte waarde -> maak dit knooppunt actief -> ga terug naar stap 2
4. Zoek gezochte waarde in actieve knooppunt op. Deze waarde wijst nu naar alle pagina's waarin gezochte waarde voorkomt -> haal al deze pagina's uit de database
5. Zoek voor elke pagina de rijen op waar de waarde gelijk is aan de gezochte waarde

De index boom bestaat uit een aantal knooppunten (**nodes**) als langwerpige rechthoeken. Het bovenste knooppunt is de **root** (wortel van de boom) en is het beginpunt van de index. Elk knooppunt bevat in bovenstaand voorbeeld maximaal 3 waarden uit de tabel (in de praktijk bevat een knooppunt echter geen 3 maar veel meer waarden). Elke waarde in een knooppunt wijst naar een ander knooppunt of een rij in de tabel. Naar elke rij in de tabel wordt door slechts één knooppunt verwezen. Een knooppunt dat naar een rij verwijst wordt een **leaf page** genoemd.

De waarden in een knooppunt zijn gesorteerd. Voor elk knooppunt (behalve de root) geldt dat de waarden in dat knooppunt altijd kleiner dan of gelijk zijn aan de waarde die naar dat knooppunt verwijst. Leaf pages zijn onderling ook gekoppeld. De indexwijzer in de leaf page verwijst naar een data page van de tabel en bestaat uit een verwijzing naar de pagina waar de rij zich bevindt en de locatie van de rij binnen deze pagina.

M.b.v. de B-tree index is het mogelijk om zonder alle rijen te doorlopen toch de gewenste rijen te vinden, waardoor het antwoord van de query aanzienlijk sneller is. De rijen van de tabel staan bij een B-tree index niet gesorteerd in het databestand. Wanneer we gegevens gesorteerd van schijf willen halen (bijvoorbeeld van waarde x tot en met waarde y) dan moeten veel pagina's meerdere malen ingelezen worden, wat de verwerkingstijd weer verslechtert.

Om gesorteerde gegevens uit het bestand te halen kunnen we beter **geclusterde indexen** gebruiken. Bij geclusterde indexen wordt de volgorde van de rijen in het databestand bepaald door de index. De

tabel bevat dan niet zoals in de tekening van boven naar beneden de volgorde  
 6,44,83,2,27,104,7,57,39,112,8,100,28,95 maar de volgorde  
 2,6,7,8,27,28,39,44,57,83,95,100,104,112. Ofwel de waarden zijn op volgorde van grootte in het databestand opgeslagen.

Als waarden in een tabel (databestand) gewijzigd worden of rijen toegevoegd of verwijderd, dan past SQL automatisch de index daarop aan, zodat de indexboom altijd in overeenstemming is met de inhoud van de tabel.

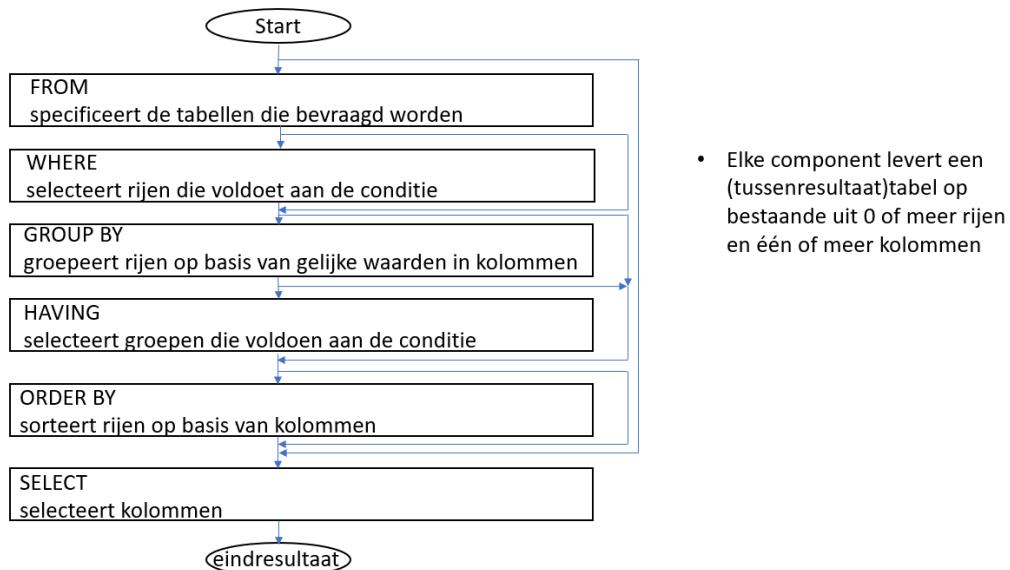
Indexes kunnen ook op combinaties van kolommen worden gedefinieerd en heten **samengestelde indexen**.

Knooppunten van indexen worden net zoals in een tabel opgeslagen in bestanden. Een index kost dus ook opslagruimte.

SQL probeert voor elke ingevoerde SQL instructie de meest efficiënte strategie te bepalen. Deze analyse heet **query optimisation** en wordt uitgevoerd door de **query optimizer**. De query optimizer is een module binnen SQL die voor elke instructie een aantal alternatieve strategieën ontwikkelt en op basis van de verwachte verwerkingstijd, aantal rijen en aanwezig indexen kiest welke strategie volgens hem het beste is. SQL voert vervolgens de instructie uit volgens de gekozen strategie.

De basisverwerkingsstrategie van een **SELECT FROM WHERE GROUP BY HAVING ORDER BY** instructie wordt in onderstaand diagram weergegeven.

## SELECT Basisverwerkingsstrategie



Bij deze **basisstrategie** wordt uitgegaan van een sequentiële zoekmethode, waarbij de tabel rij voor rij wordt doorlopen. Er wordt in de basisstrategie dus eerst alle rijen uit de pages van het databestand gelezen, dan wordt er geselecteerd welke rijen in het resultaat moeten komen, dan wordt er gegroepeerd en daarvan bepaald wat in de groepering moet komen, dan worden de rijen gesorteerd en tot slot worden de gewenste kolommen uit de geselecteerde rijen opgehaald.

Door het gebruik van indexen wordt deze basisstrategie veranderd in een **geoptimaliseerde strategie**. Naarmate de instructies ingewikkelder worden, wordt het ook voor de query optimizer moeilijker de optimale verwerkingsstrategie te bepalen waardoor de verwerkingstijd toeneemt.

### 25.2.1 Creëren van Indexen

Indexen kunnen worden aangemaakt met de syntax:

```
CREATE <indextype> INDEX <indexnaam>
ON <tabelspecificatie>
(<kolom-in-index> [,<kolom-in-index>]...)
```

waarbij:

```
<indextype> = UNIQUE | CLUSTERED
<kolom-in-index> = <kolomnaam> [ASC | DESC]
```

Wanneer er geen <indextype> wordt opgegeven dan wordt er een niet-unieke index (**non-UNIQUE index**) gecreëerd.

Wanneer in een SELECT instructie op een kolom aflopend gesorteerd wordt, is de verwerkingstijd korter wanneer op die kolom ook een aflopende index gedefinieerd is met DESC.

Meerdere kolommen mogen bij de definitie van en index opgenomen worden, zolang ze maar allemaal tot dezelfde tabel behoren.

Na het toevoegen van een **UNIQUE index** zorgt SQL ervoor dat er nooit twee gelijke waarden/combinaties ingevoerd kunnen worden. Het is niet mogelijk om een unieke index op een gevulde tabel aan te maken als de betreffende kolom reeds dubbele waarden bevat. De dubbele waarden moeten dan eerst uit de tabel worden verwijderd.

Voor sommige SQL producten (waaronder SQL Server) geldt dat een kolom waarop een unieke index gedefinieerd is maximaal één nul-waarde kan bevatten. Een kolom waarop een niet-unieke index gedefinieerd is kan meerdere nul-waarden bevatten.

Indexen kunnen ook met een ALTER TABLE worden toegevoegd met de syntax:

```
ALTER TABLE <tabelspecificatie>
ADD <indextype> INDEX <indexnaam>
(<kolom-in-index> [,<kolom-in-index>])
```

waarbij:

```
<indextype> = UNIQUE | CLUSTERED
<kolom-in-index> = <kolomnaam> [ASC | DESC]
```

Een index kan worden verwijderd met de syntax:

```
DROP INDEX <tabelnaam>.<indexnaam>
```

Veel SQL producten (waaronder SQL Server) creëren automatisch een unieke index als een primaire of alternatieve sleutel met een CREATE TABLE of ALTER TABLE instructie gedefinieerd wordt. De naam van de index wordt door het SQL product zelf bepaald.

Het effect van een index wordt pas duidelijk als een tabel een groot aantal records bevat. Het effect van het gebruik van een index verschilt erg veel of er tientallen of duizenden rijen in de tabel zit.

De index levert alleen tijdwinst op als het aantal rijen dat geselecteerd wordt een klein percentage van het aantal rijen in de tabel is.

```
Procedure werkt nog niet!!!
```

```
USE TENNIS;
CREATE TABLE SPELERS_XXL (
SPELERSNR    INT NOT NULL PRIMARY KEY,
NAAM        CHAR(15) NOT NULL,
VOORLETTERS  CHAR(3) NOT NULL,
GEB_DATUM    DATE,
GESLACHT     CHAR(1) NOT NULL,
JAARTOE      SMALLINT NOT NULL,
STRAAT       VARCHAR(30) NOT NULL,
HUISNR       CHAR(4),
POSTCODE      CHAR(6),
PLAATS       VARCHAR(30) NOT NULL,
TELEFOON     CHAR(13),
BONDNR       CHAR(8));

CREATE PROCEDURE VUL_SPELERS_XXL
(IN AANTAL_SPELERS INTEGER)
BEGIN
    DECLARE TELLER INTEGER;
    TRUNCATE TABLE SPELERS_XXL;
    COMMIT WORK;
    SET TELLER = 1;
    WHILE TELLER <= AANTAL_SPELERS DO
        INSERT INTO SPELERS_XXL VALUES(
            TELLER,
            CONCAT('naam',CAST(TELLER AS CHAR(10))),
            CASE MOD(TELLER,2) WHEN 0 THEN 'v11' ELSE 'v12' END,
            DATE('1960-01-01') + INTERVAL (MOD(TELLER,300)) MONTH,
            CASE MOD(TELLER,20) WHEN 0 THEN 'V'ELSE 'M' END,
            1980 + MOD(TELLER,20),
            CONCAT('straat',CAST(TELLER/10 AS UNASIGNED INTEGER)),
            CAST(CAST(TELLER/10 AS UNASIGNED INTEGER)+1 AS CHAR(4)),
            CONCAT('p',MOD(TELLER,50)),
            CONCAT('plaats',MOD(TELLER,10)),
            '070-6868689',
            CASE MOD TELLER(TELLER,3) WHEN 0
                THEN NULL ELSE CAST(TELLER AS CHAR(8)) END);
        IF MOD(TELLER,1000 = 0 THEN
            COMMIT WORK;
        END IF;
        SET TELLER = TELLER + 1;
    END WHILE;
    COMMIT WORK;
END

CALL VUL_SPELERS_XXL(100000)

CREATE INDEX SPELERS_XXL_VOORLETTERS
ON SPELERS_XXL(VOORLETTERS)

CREATE INDEX SPELERS_XXL_POSTCODE
```

```
ON SPELERS_XXL(POSTCODE)
```

```
CREATE INDEX SPELERS_XXL_STRAAT
ON SPELERS_XXL(PLAATS)
```

Om alle SELECT instructies op een tabel gebruik te laten maken van een index zouden we op elke kolom en op elke combinatie van kolommen een index moeten creëren. Echter kost elke index opslagruimte en moet bij elke update alle indexen gewijzigd worden, wat weer vertragend werkt. Daarom moet er een keuze worden gemaakt op welke kolommen of kolomcombinaties we indexen maken. Daarvoor kunnen we een aantal richtlijnen gebruiken:

- Definieer op elke kandidaatssleutel een index zodat de uniciteit van nieuwe waarden snel gecontroleerd kan worden. Bij de meeste SQL producten (waaronder SQL Server) wordt bij het aanmaken van een primary of kandidaatssleutel automatisch een unique index aangemaakt.
- Definieer op elke refererende sleutel een index
- Definieer een index op kolommen waarop geselecteerd wordt  
Bij een query `SELECT * FROM <tabelnaam> WHERE <kolomnaam> = <waarde>`; een index plaatsen op de kolom `<kolomnaam>`  
Dit geldt ook indien er in plaats van de `=` een `<, <=, >, >=` wordt gebruikt maar NIET voor `<>`

Ook hier geldt dat de index alleen tijdwinst oplevert als het aantal rijen dat geselecteerd wordt een klein percentage van het aantal rijen in de tabel is.

Of een index veel tijdwinst oplevert hangt af van een aantal factoren, zoals het aantal rijen in de tabel (Cardinaliteit van de tabel), het aantal verschillende waarden in de betreffende kolom (Cardinaliteit van de kolom) en de distributie van de waarden binnen de kolom:

- Hoe groter de tabel, des te groter de invloed van de index. Of een tabel groot genoeg is voor een index is afhankelijk van het systeem waarop het draait, en zal uitgeprobeerd moeten worden
- Het toevoegen van een index op een kolom met een lage cardinaliteit (weinig verschillende waarden) heeft weinig invloed. Als de tabel groter wordt dan begint de verwerkingssnelheid beter te worden maar blijft minimaal.
- Het effect van het toevoegen van een index op een kolom hangt ook af van de distributie van de waarden binnen de kolom. Als bijvoorbeeld geslacht 95% M is en 5% V, dan zal het tellen van de vrouwen (`SELECT COUNT(*) FROM <tabelnaam> WHERE GESLACHT = 'V'`; aanzienlijk veel tijdwinst opleveren).
- Als een WHERE component een AND operatie bevat, kan het definiëren van een index op een combinatie van kolommen tot een efficiënte verwerking leiden.
- Wanneer er geen index op een tabel is en er op kolommen van de tabel wordt gesorteerd, dan moet een aparte (tijdrovende) sorteerslag uitgevoerd worden. Deze extra sorteerslag kan voorkomen worden door een index op de desbetreffende kolom te definiëren. Bij het ophalen van rijen uit de database wordt de index gebruikt. Het tussenresultaat is dus reeds gesorteerd en de extra sorteerslag is niet meer nodig. Dit geldt alleen als de betreffende kolom weinig NULL waarden bevat, omdat NULL waarden niet in de index worden opgeslagen en als de instructie geen WHERE component heeft met een conditie die goed te optimaliseren is. SQL moet een sortering uitvoeren wanneer er een ORDER BY wordt

gebruikt, maar ook bij een GROUP BY en een DISTINCT moeten de rijen eerst gesorteerd worden en heeft de index voordeel dat de gegevens reeds gesorteerd zijn.

### 25.2.2 Execution Plan

In onderstaand voorbeeld gaan we uit van de volgende tabel tblTable met 9 rijen:

1	Object	Color	ID
2	Table	Brown	1
3	Table	Black	2
4	Computer	Gold	3
5	Printer	Black	4
6	Printer	Red	5
7	Bookcase	Brown	6
8	Computer	Black	7
9	Bookcase	White	8
10	Book	Green	9

Wanneer we op deze tabel uitvoeren `SELECT * FROM tblTable WHERE Object = 'Printer'`; dan loopt SQL elke waarde in de kolom Object langs om te kijken of de waarde voldoet aan 'Printer'. Hij moet daarvoor een (full table) scan uitvoeren. En dat is niet erg efficiënt als je niet 9 rijen maar 9 miljard rijen hebt. Hij gaat dan al de 9 miljard rijen langs. Daarvoor worden nu indexen gebruikt.

Er zijn twee hoofdtypen indexen:

- **Clustered index**

Een clustered index sorteert de gegevens in de tabel in de volgorde van de index. De index moet unique zijn. Het is dus niet mogelijk om een clustered index te maken op de kolom Object omdat die niet uniek is. Het is ook niet mogelijk om een clustered index te maken op de kolom Color omdat die ook niet uniek is. Het is wel mogelijk om een clustered index op de kolom ID te maken omdat die wel uniek is. Het is ook wel mogelijk om een clustered index te maken op de kolommen Object en Color omdat de combinatie van die twee wel uniek is. Het is ook mogelijk om een clustered index te maken op de kolommen Color en Object. Het verschil tussen de laatste twee is dat clustered indexen fysiek de tabel sorteren, dus een sortering op Object en Color geeft een andere volgorde dan sortering op Color en Object. De keuze welke het beste te gebruiken is hangt af van de soort queries die je wilt uitvoeren. De index gebruikt eigenlijk een B-tree. Als je een query gebruikt als `SELECT * FROM tblTable WHERE Object = 'Printer'`; dan is het niet efficiënt als je de sortering gebruikt die gebaseerd is op de kolom Color. In dat geval zal het meerder searches moeten uitvoeren. Een search voor de zwarte kleur, dan een search voor de bruine kleur, etc. maar binnen elke search. Dit heet daarom een **clustered search**.

- **Non-clustered index**

Een non-clustered index sorteert de gegevens in de tabel niet, maar creëert een aparte index.

	A	B	C
1	Object	Color	ID
2	Table	Black	1
3	Printer	Black	2
4	Computer	Black	3
5	Table	Brown	4
6	Bookcase	Brown	5
7	Computer	Gold	6
8	Printer	Red	7
9	Bookcase	White	8
10	Book	Green	9

Dus als ik een index heb op Color en hij kijkt naar de psuedo values aan de linkerzijde, dan zou hij zeggen black is 1,2,3 brown = 4,5 etc.

### 25.3 Optimaliseren van instructies

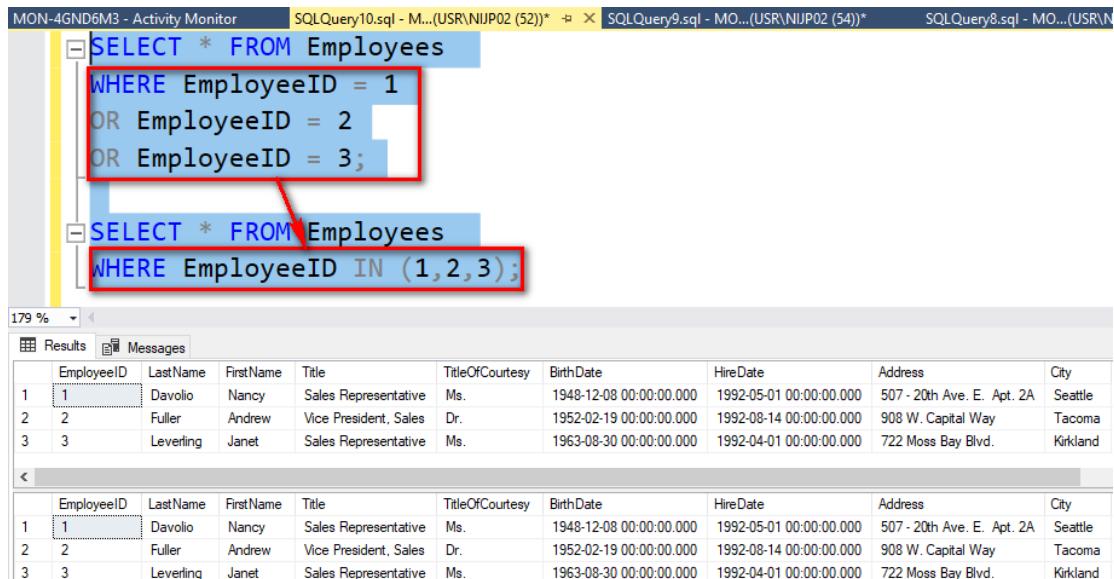
Zoals hierboven aangegeven kunnen indexen de verwerkingstijd van instructies aanzienlijk versnellen, echter kan de Query Optimizer niet altijd de snelste verwerkingsstrategie bepalen en kiest de Query Optimizer ondanks de aanwezigheid van een index soms toch voor een sequentiële verwerkingsstrategie.

Daarom volgt in deze paragraaf een aantal richtlijnen om de SQL instructies zodanig te optimaliseren dat de Query Optimizer wel zal kiezen voor het gebruik van indexen.

Het is niet noodzakelijk om alle richtlijnen uit het hoofd te kennen, maar het is wel goed om er globaal kennis van te hebben en zo nodig in deze paragraaf of in online documentatie terug te zoeken wanneer dat nodig is.

- Vermijd de OR operatie door deze te vervangen door een IN of een UNION operatie

IN:



```

SELECT * FROM Employees
WHERE EmployeeID = 1
OR EmployeeID = 2
OR EmployeeID = 3;

SELECT * FROM Employees
WHERE EmployeeID IN (1, 2, 3);

```

EmployeeID	Lastname	Firstname	Title	TitleOfCourtesy	BirthDate	HireDate	Address	City
1	Davolio	Nancy	Sales Representative	Ms.	1948-12-08 00:00:00.000	1992-05-01 00:00:00.000	507 - 20th Ave. E. Apt. 2A	Seattle
2	Fuller	Andrew	Vice President, Sales	Dr.	1952-02-19 00:00:00.000	1992-08-14 00:00:00.000	908 W. Capital Way	Tacoma
3	Leverling	Janet	Sales Representative	Ms.	1963-08-30 00:00:00.000	1992-04-01 00:00:00.000	722 Moss Bay Blvd.	Kirkland

### UNION:

MON-4GND6M3 - Activity Monitor    SQLQuery10.sql - M... (USR\NIJP02 (52)) \* → X SQLQuery9.sql - MO... (USR\NIJP02 (54)) \*    SQLQuery8.sql - MO... (USR\NIJP02 (53)) \*

```

SELECT EmployeeID, FirstName, LastName, Title, City FROM Employees
WHERE Title = 'Inside Sales Coordinator'
OR City = 'London';

SELECT EmployeeID, FirstName, LastName, Title, City FROM Employees
WHERE Title = 'Inside Sales Coordinator'
UNION
SELECT EmployeeID, FirstName, LastName, Title, City FROM Employees
WHERE City = 'London';

```

179 %

	EmployeeID	FirstName	LastName	Title	City
1	5	Steven	Buchanan	Sales Manager	London
2	6	Michael	Suyama	Sales Representative	London
3	7	Robert	King	Sales Representative	London
4	8	Laura	Callahan	Inside Sales Coordinator	Seattle
5	9	Anne	Dodsworth	Sales Representative	London

	EmployeeID	FirstName	LastName	Title	City
1	5	Steven	Buchanan	Sales Manager	London
2	6	Michael	Suyama	Sales Representative	London
3	7	Robert	King	Sales Representative	London
4	8	Laura	Callahan	Inside Sales Coordinator	Seattle
5	9	Anne	Dodsworth	Sales Representative	London

- Vermijd onnodig gebruik van UNION operatie

De UNION kan hierboven gebruikt worden in plaats van de OR zodat de Query Optimizer toch de index gebruikt. We moeten bij het gebruik van een UNION echter wel oppassen of we daardoor niet verzorgen dat twee maal de gehele tabel wordt doorlopen. Bijvoorbeeld als we de absolute waarde willen weten van de verschillen tussen de artikelen in voorraad en het voorraadniveau waarbij er opnieuw een bestelling moet worden gedaan om de voorraad aan te vullen, dan kunnen we beter gebruik maken van de ABS functie.

MON-4GND6M3 - Activity Monitor    SQLQuery10.sql - M...(USR\NIJP02 (52))\*    SQLQuery9.sql - MO...(USR\NIJP02 (54))\*    SQLQuery

```

SELECT ProductID, UnitsInStock - ReorderLevel
FROM Products
WHERE UnitsInStock > ReorderLevel
UNION
SELECT ProductID, ReorderLevel - UnitsInStock
FROM Products
WHERE UnitsInStock <= ReorderLevel;

```

↓

```

SELECT ProductID, ABS(UnitsInStock - ReorderLevel)
FROM Products;

```

179 %

	ProductID	(No column name)
57	57	16
58	58	42
59	59	79
60	60	19
	ProductID	(No column name)
60	60	19
61	61	88
62	62	17
63	63	19

- Vermijd de NOT operation  
Wanneer de WHERE een NOT bevat dan zal de Query Optimizer meestal geen index gebruiken. Vervang daarvoor zo mogelijk de NOT operator door vergelijkningsoperatoren

MON-4GND6M3 - Activity Monitor    SQLQuery10.sql - M...(USR\NIJP02 (52))\*    SQLQuery9.sql - MO...(USR\NIJP02 (54))\*    SQLQuery8.sql - MO...(USR\NIJP02 (54))\*

```

SELECT * FROM Orders
WHERE NOT OrderDate > '1997';

```

↓

```

SELECT * FROM Orders
WHERE OrderDate <= '1997';

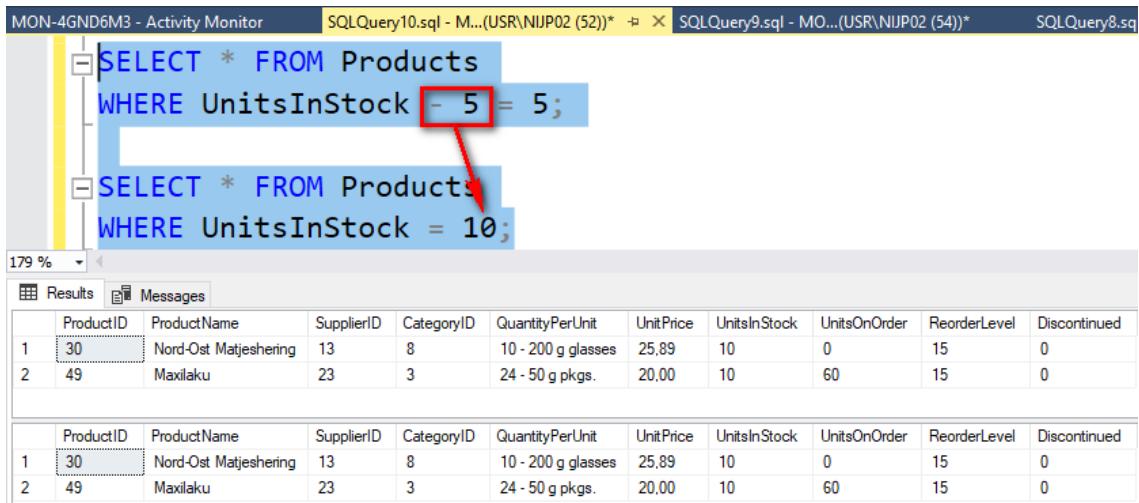
```

179 %

	OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate	ShippedDate	ShipVia	Freight	ShipName
1	10248	VINET	5	1996-07-04 00:00:00.000	1996-08-01 00:00:00.000	1996-07-16 00:00:00.000	3	32,38	Vins et alcools Chevalier
2	10249	TOMSP	6	1996-07-05 00:00:00.000	1996-08-16 00:00:00.000	1996-07-10 00:00:00.000	1	11,61	Toms Spezialitäten
3	10250	HANAR	4	1996-07-08 00:00:00.000	1996-08-05 00:00:00.000	1996-07-12 00:00:00.000	2	65,83	Hanari Cames
4	10251	VICTE	3	1996-07-08 00:00:00.000	1996-08-05 00:00:00.000	1996-07-15 00:00:00.000	1	41,34	Victuailles en stock
5	10252	SUPRD	4	1996-07-09 00:00:00.000	1996-08-06 00:00:00.000	1996-07-11 00:00:00.000	2	51,30	Suprèmes délices
	OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate	ShippedDate	ShipVia	Freight	ShipName
1	10248	VINET	5	1996-07-04 00:00:00.000	1996-08-01 00:00:00.000	1996-07-16 00:00:00.000	3	32,38	Vins et alcools Chevalier
2	10249	TOMSP	6	1996-07-05 00:00:00.000	1996-08-16 00:00:00.000	1996-07-10 00:00:00.000	1	11,61	Toms Spezialitäten
3	10250	HANAR	4	1996-07-08 00:00:00.000	1996-08-05 00:00:00.000	1996-07-12 00:00:00.000	2	65,83	Hanari Cames
4	10251	VICTE	3	1996-07-08 00:00:00.000	1996-08-05 00:00:00.000	1996-07-15 00:00:00.000	1	41,34	Victuailles en stock
5	10252	SUPRD	4	1996-07-09 00:00:00.000	1996-08-06 00:00:00.000	1996-07-11 00:00:00.000	2	51,30	Suprèmes délices

- Isoleer kolommen in condities.  
Wanneer de kolom waarop de index gedefinieerd is gebruikt wordt binnen een scalaire functie dan wordt de index niet gebruikt. Dit kan opgelost worden door de kolom aan de

linkerzijde van het = teken te isoleren, ofwel de constante naar de andere kant te brengen



```

SELECT * FROM Products
WHERE UnitsInStock = 5;

SELECT * FROM Products
WHERE UnitsInStock = 10;
    
```

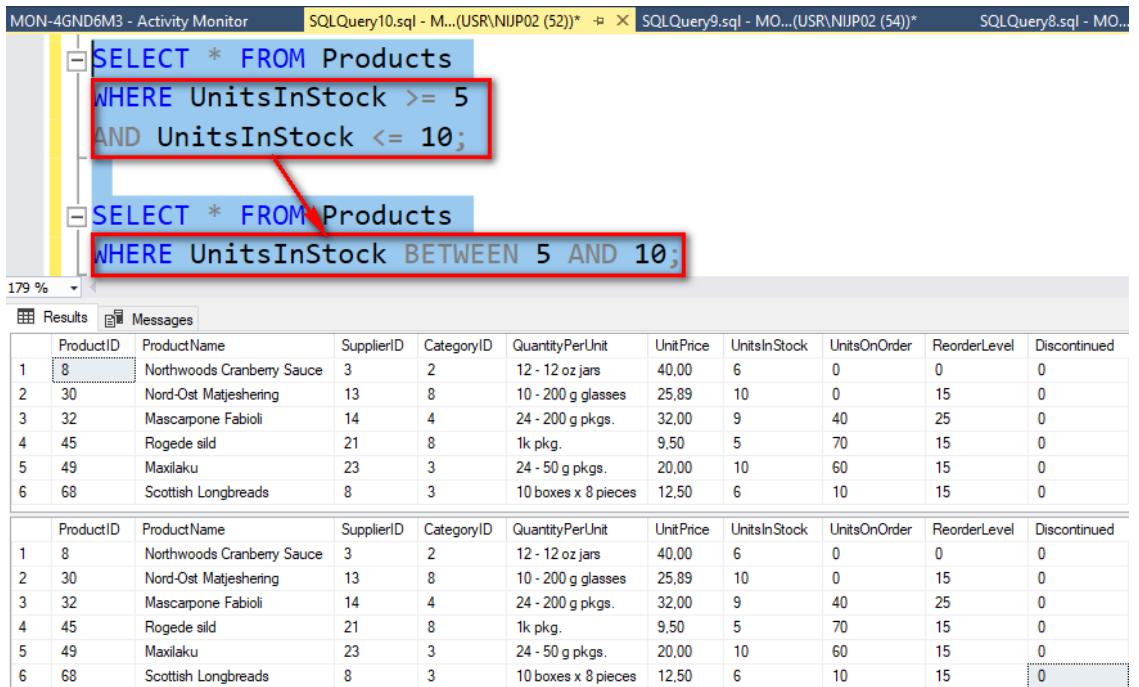
ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel	Discontinued	
1	30	Nord-Ost Matjeshering	13	8	10 - 200 g glasses	25.89	10	0	15	0
2	49	Maxilaku	23	3	24 - 50 g pkgs.	20.00	10	60	15	0

ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel	Discontinued	
1	30	Nord-Ost Matjeshering	13	8	10 - 200 g glasses	25.89	10	0	15	0
2	49	Maxilaku	23	3	24 - 50 g pkgs.	20.00	10	60	15	0

- Gebruik de BETWEEN operator

Wanneer een AND wordt gebruikt om te zoeken naar waarden die in een bepaald bereik voorkomen dan zal er meestal geen index worden gebruikt. Dit kan voorkomen worden door BETWEEN te gebruiken.



```

SELECT * FROM Products
WHERE UnitsInStock >= 5
AND UnitsInStock <= 10;

SELECT * FROM Products
WHERE UnitsInStock BETWEEN 5 AND 10;
    
```

ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel	Discontinued	
1	8	Northwoods Cranberry Sauce	3	2	12 - 12 oz jars	40.00	6	0	0	0
2	30	Nord-Ost Matjeshering	13	8	10 - 200 g glasses	25.89	10	0	15	0
3	32	Mascarpone Fabioli	14	4	24 - 200 g pkgs.	32.00	9	40	25	0
4	45	Rogede sild	21	8	1k pkg.	9.50	5	70	15	0
5	49	Maxilaku	23	3	24 - 50 g pkgs.	20.00	10	60	15	0
6	68	Scottish Longbreads	8	3	10 boxes x 8 pieces	12.50	6	10	15	0

ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel	Discontinued	
1	8	Northwoods Cranberry Sauce	3	2	12 - 12 oz jars	40.00	6	0	0	0
2	30	Nord-Ost Matjeshering	13	8	10 - 200 g glasses	25.89	10	0	15	0
3	32	Mascarpone Fabioli	14	4	24 - 200 g pkgs.	32.00	9	40	25	0
4	45	Rogede sild	21	8	1k pkg.	9.50	5	70	15	0
5	49	Maxilaku	23	3	24 - 50 g pkgs.	20.00	10	60	15	0
6	68	Scottish Longbreads	8	3	10 boxes x 8 pieces	12.50	6	10	15	0

- Vermijd bepaalde vormen van de LIKE operator.

Wanneer een LIKE wordt gebruikt waarbij het patroon begint met een % teken dan wordt er geen index gebruikt

MON-4GND6M3 - Activity Monitor    SQLQuery10.sql - M... (USR\NJP02 (52)) \*    SQLQuery9.sql - MO... (USR\NJP02 (54)) \*    SQLQuery8.sql - MO...

```
SELECT * FROM EMPLOYEES WHERE Title LIKE '%sales';
```

179 %

	EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	BirthDate	HireDate	Address	City
1	2	Fuller	Andrew	Vice President, Sales	Dr.	1952-02-19 00:00:00.000	1992-08-14 00:00:00.000	908 W. Capital Way	Tacoma

- Voeg bij JOINS redundante condities toe  
Joins zijn soms eenvoudig te versnellen door aan de WHERE component een extra conditie toe te voegen die het eindresultaat niet veranderd.

MON-4GND6M3 - Activity Monitor    SQLQuery10.sql - M... (USR\NJP02 (52)) \*    SQLQuery9.sql - MO... (USR\NJP02 (54)) \*    SQLQuery8.sql - MO...

```
SELECT *
FROM Orders o, Employees e
WHERE e.EmployeeID = o.EmployeeID
AND o.EmployeeID = 5;

SELECT *
FROM Orders o, Employees e
WHERE e.EmployeeID = o.EmployeeID
AND o.EmployeeID = 5
AND e.EmployeeID = 5;
```

179 %

	OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate	ShippedDate	ShipVia	Freight	ShipName
1	10248	VINET	5	1996-07-04 00:00:00.000	1996-08-01 00:00:00.000	1996-07-16 00:00:00.000	3	32.38	Vins et alcools Chevalier
2	10254	CHOPS	5	1996-07-11 00:00:00.000	1996-08-08 00:00:00.000	1996-07-23 00:00:00.000	2	22.98	Chop-suey Chinese
3	10269	WHITC	5	1996-07-31 00:00:00.000	1996-08-14 00:00:00.000	1996-08-09 00:00:00.000	1	4.56	White Clover Markets
4	10297	BLONP	5	1996-09-04 00:00:00.000	1996-10-16 00:00:00.000	1996-09-10 00:00:00.000	2	5.74	Blondel père et fils
5	10320	WARTH	5	1996-10-03 00:00:00.000	1996-10-17 00:00:00.000	1996-10-18 00:00:00.000	3	34.57	Wartian Herkku

	OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate	ShippedDate	ShipVia	Freight	ShipName
1	10248	VINET	5	1996-07-04 00:00:00.000	1996-08-01 00:00:00.000	1996-07-16 00:00:00.000	3	32.38	Vins et alcools Chevalier
2	10254	CHOPS	5	1996-07-11 00:00:00.000	1996-08-08 00:00:00.000	1996-07-23 00:00:00.000	2	22.98	Chop-suey Chinese
3	10269	WHITC	5	1996-07-31 00:00:00.000	1996-08-14 00:00:00.000	1996-08-09 00:00:00.000	1	4.56	White Clover Markets
4	10297	BLONP	5	1996-09-04 00:00:00.000	1996-10-16 00:00:00.000	1996-09-10 00:00:00.000	2	5.74	Blondel père et fils
5	10320	WARTH	5	1996-10-03 00:00:00.000	1996-10-17 00:00:00.000	1996-10-18 00:00:00.000	3	34.57	Wartian Herkku

- Vermijd de HAVING component  
Probeer zo veel mogelijk condities in de WHERE component te plaatsen omdat de condities in de HAVING geen index gebruiken

MON-4GND6M3 - Activity Monitor    SQLQuery10.sql - M... (USR\NJP02 (52)) \*    SQLQuery9.sql - MO... (USR\NJP02 (54)) \*    SQLQuery8.sql - MO...

```
SELECT OrderID, COUNT(*)
FROM Orders
GROUP BY OrderID
HAVING OrderID < 10255;

SELECT OrderID, COUNT(*)
FROM Orders
WHERE OrderID < 10255
GROUP BY OrderID;
```

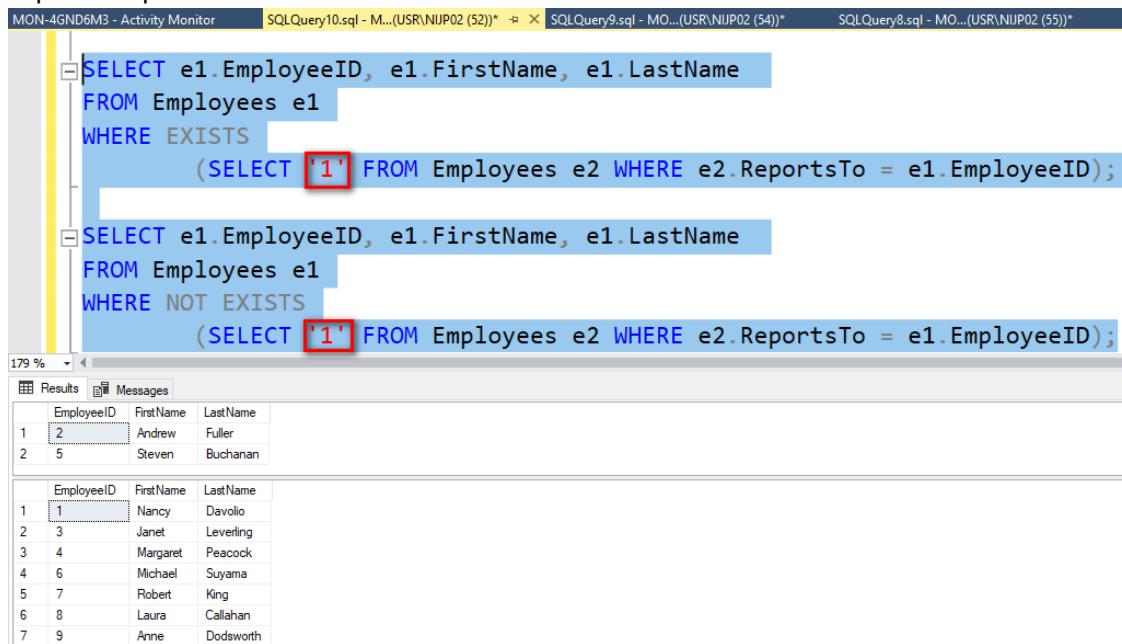
Results		Messages	
	OrderID	(No column name)	
1	10251	1	
2	10253	1	
3	10250	1	
4	10252	1	
5	10248	1	
6	10254	1	
7	10249	1	

	OrderID	(No column name)	
1	10251	1	
2	10253	1	
3	10250	1	
4	10252	1	
5	10248	1	
6	10254	1	
7	10249	1	

- Maak de SELECT component zo smal mogelijk

In de SELECT van de hoofdquery wordt geformuleerd welke gegevens gepresenteerd moeten worden. Probeer te voorkomen dat onnodige kolommen gepresenteerd worden. Dit kan een nadelige invloed op de verwerkingsnelheid hebben. Wanneer de SELECT van de subquery m.b.v. een EXISTS gekoppeld is met de hoofdquery, dan maakt het niet uit wat er in de subquery geselecteerd wordt omdat het geen effect heeft op het eindresultaat van de SELECT instructie van de hoofdquery. Neem daarom in de SELECT van de subquery één expressie op met één constante.



```

SELECT e1.EmployeeID, e1.FirstName, e1.LastName
FROM Employees e1
WHERE EXISTS
    (SELECT '1' FROM Employees e2 WHERE e2.ReportsTo = e1.EmployeeID);

SELECT e1.EmployeeID, e1.FirstName, e1.LastName
FROM Employees e1
WHERE NOT EXISTS
    (SELECT '1' FROM Employees e2 WHERE e2.ReportsTo = e1.EmployeeID);

```

	EmployeeID	FirstName	LastName
1	2	Andrew	Fuller
2	5	Steven	Buchanan

	EmployeeID	FirstName	LastName
1	1	Nancy	Davolio
2	3	Janet	Leverling
3	4	Margaret	Peacock
4	6	Michael	Suyama
5	7	Robert	King
6	8	Laura	Callahan
7	9	Anne	Dodsworth

- Vermijd DISTINCT

Het verwijderen van dubbele rijen uit een resultaat m.b.v. DISTINCT in de SELECT kan leiden tot een lange verwerkingsstijd. Vermijd daarom DISTINCT als het niet echt noodzakelijk of zelfs overbodig is. De Distinct in onderstaand voorbeeld is overbodig omdat OrderID de primaire sleutel is van de tabel Orders en de waarden dus al uniek zijn.

MON-4GND6M3 - Activity Monitor    SQLQuery10.sql - M...(USR\NIJP02 (52))\*    X SQLQuery9.sql - MO...(USR\NIJP02 (54))\*    SQLQuery8.sql - MO...(USR\NIJP02 (55))\*

```

SELECT o.OrderID
  FROM Orders o, Employees e
 WHERE o.EmployeeID = e.EmployeeID;

SELECT DISTINCT o.OrderID
  FROM Orders o, Employees e
 WHERE o.EmployeeID = e.EmployeeID;
  
```

179 %

	OrderID
1	10251
2	10253
3	10256
4	10266
5	10273

	OrderID
1	10251
2	10253
3	10256
4	10266
5	10273

- Gebruik de ALL optie bij SET operatoren  
ALL zorgt ervoor dat er (achter de schermen) niet gesorteerd wordt om dubbele rijen te verwijderen.

MON-4GND6M3 - Activity Monitor    SQLQuery10.sql - M...(USR\NIJP02 (52))\*    X SQLQuery9.sql - MO...(USR\NIJP02 (54))\*    SQLQuery8.sql - MO...(USR\NIJP02 (55))\*

```

SELECT * FROM Employees;

SELECT FirstName, LastName
  FROM Employees
 WHERE City = 'London'
UNION ALL
SELECT FirstName, LastName
  FROM Employees
 WHERE City = 'Seattle'
  
```

179 %

EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	BirthDate	HireDate	Address	City
1	Davolio	Nancy	Sales Representative	Ms.	1948-12-08 00:00:00.000	1992-05-01 00:00:00.000	507 - 20th Ave. E. Apt. 2A	Seattle
2	Fuller	Andrew	Vice President, Sales	Dr.	1952-02-19 00:00:00.000	1992-08-14 00:00:00.000	908 W. Capital Way	Tacoma
3	Leverling	Janet	Sales Representative	Ms.	1963-08-30 00:00:00.000	1992-04-01 00:00:00.000	722 Moss Bay Blvd.	Kirkland
4	Peacock	Margaret	Sales Representative	Mrs.	1937-09-19 00:00:00.000	1993-05-03 00:00:00.000	4110 Old Redmond Rd.	Redmond
5	Buchanan	Steven	Sales Manager	Mr.	1955-03-04 00:00:00.000	1993-10-17 00:00:00.000	14 Garrett Hill	London
6	Suyama	Michael	Sales Representative	Mr.	1963-07-02 00:00:00.000	1993-10-17 00:00:00.000	Coventry House Miner Rd.	London
7	King	Robert	Sales Representative	Mr.	1960-05-29 00:00:00.000	1994-01-02 00:00:00.000	Edgeham Hollow Winchester Way	London
8	Callahan	Laura	Inside Sales Coordinator	Ms.	1958-01-09 00:00:00.000	1994-03-05 00:00:00.000	4726 - 11th Ave. N.E.	Seattle
9	Dodsworth	Anne	Sales Representative	Ms.	1966-01-27 00:00:00.000	1994-11-15 00:00:00.000	7 Hounds tooth Rd.	London

FirstName	LastName
1 Steven	Buchanan
2 Michael	Suyama
3 Robert	King
4 Anne	Dodsworth
5 Nancy	Davolio
6 Laura	Callahan

Er wordt nu geen sorteerslag uitgevoerd om dubbele rijen te voorkomen. Deze sorteerslag zou onnodig zijn daar elke speler maar in één plaats woont

- Geef voorkeur aan OUTEr-joins boven UNION operatoren  
OUTER JOIN is binnen SQL een moderne techniek voor problemen die voordien met een UNION werden opgelost.

Bijvoorbeeld: een overzicht van alle Customers met hun orders gesorteerd op customers

SQLQuery27.sql - M... (USR\NIJP02 (65)) \* → X SQLQuery26.sql - M... (USR\NIJP02 (52)) \*

```

SELECT Customers.CustomerID, CompanyName, OrderID
  FROM Customers, Orders
 WHERE Customers.CustomerID = Orders.CustomerID
UNION
SELECT CustomerID, CompanyName, NULL
  FROM Customers
 WHERE CustomerID NOT IN
      (SELECT CustomerID FROM Orders)
 ORDER BY CustomerID;

SELECT Customers.CustomerID, CompanyName, OrderID
  FROM Customers LEFT OUTER JOIN Orders
    ON (Customers.CustomerID = Orders.CustomerID)
 ORDER BY CustomerID;

```

	CustomerID	CompanyName	OrderID
1	ALFKI	Alfreds Futterkiste	10643
2	ALFKI	Alfreds Futterkiste	10692
3	ALFKI	Alfreds Futterkiste	10702
4	ALFKI	Alfreds Futterkiste	10835
5	ALFKI	Alfreds Futterkiste	10952
6	ALFKI	Alfreds Futterkiste	11011
7	ANATR	Ana Trujillo Emparedados y helados	10308
8	ANATR	Ana Trujillo Emparedados y helados	10625
9	ANATR	Ana Trujillo Emparedados y helados	10759
	CustomerID	CompanyName	OrderID
1	ALFKI	Alfreds Futterkiste	10643
2	ALFKI	Alfreds Futterkiste	10692
3	ALFKI	Alfreds Futterkiste	10702
4	ALFKI	Alfreds Futterkiste	10835
5	ALFKI	Alfreds Futterkiste	10952
6	ALFKI	Alfreds Futterkiste	11011
7	ANATR	Ana Trujillo Emparedados y helados	10308
8	ANATR	Ana Trujillo Emparedados y helados	10625
9	ANATR	Ana Trujillo Emparedados y helados	10759

- Vermijd datatype conversie  
SQL voert zo nodig automatisch data conversies uit.  
Bijvoorbeeld: **WHERE EmployeeID = '2'** terwijl EmployeeID datatype INT heeft.  
Deze conditie werkt correct, ook al wordt hierbij een INT vergeleken met een alfanumerieke constante. Dit converteren heeft een nadelige invloed op de verwerkingsnelheid. Als dergelijke vergelijkingen niet noodzakelijk zijn probeer ze dan te vermijden.

- De grootste tabel als laatste

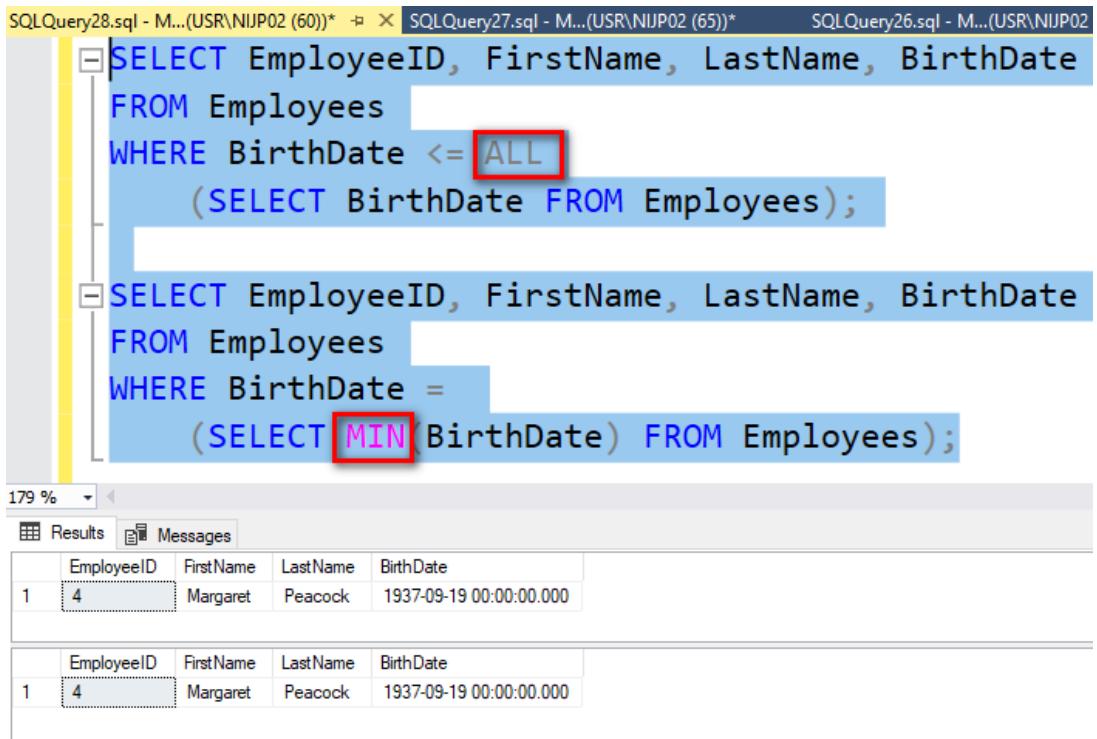
Bij het formuleren van JOINS kan de volgorde van de tabellen in de FROM component effect hebben op de verwerkingsnelheid. Plaats daarom de grootste tabel als laatste in de FROM component.

Bijvoorbeeld: Als de tabel Orders groter is dan de tabel Employees, dan:

**FROM Employees, Orders** in plaats van **FROM Orders, Employees**

- Vermijd ANY en ALL operatoren

Sommige optimizers zullen voor het verwerken van condities met de ALL operator geen index gebruiken. Vervang daarom zo mogelijk een ALL operator door een aggregatiefunctie MIN of MAX.



```

SQLQuery28.sql - M...(USR\NIJP02 (60))*
SQLQuery27.sql - M...(USR\NIJP02 (65))*
SQLQuery26.sql - M...(USR\NIJP02

SELECT EmployeeID, FirstName, LastName, BirthDate
FROM Employees
WHERE BirthDate <= ALL
    (SELECT BirthDate FROM Employees);

SELECT EmployeeID, FirstName, LastName, BirthDate
FROM Employees
WHERE BirthDate =
    (SELECT MIN BirthDate) FROM Employees);

```

	EmployeeID	FirstName	LastName	BirthDate
1	4	Margaret	Peacock	1937-09-19 00:00:00.000

	EmployeeID	FirstName	LastName	BirthDate
1	4	Margaret	Peacock	1937-09-19 00:00:00.000

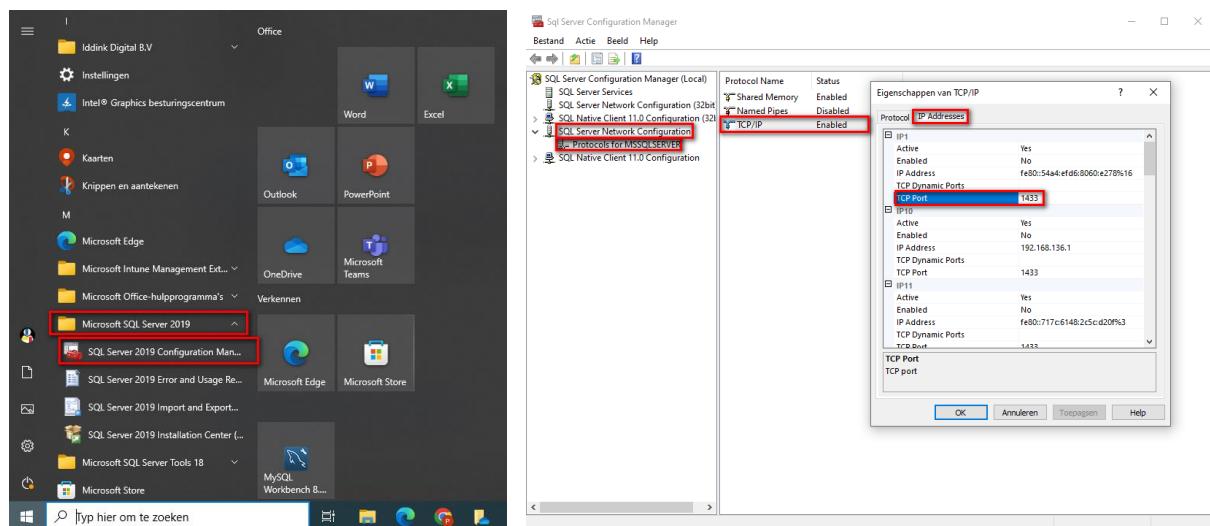
## 25.4 Indexing and Performance (Admin Guide)

De SQL Server architectuur heeft vier hoofdcomponenten:

UTILITIES DBCC, BCP, SQLCMD	PROTOCOLS
	TCP/IP, Shared Memory, Named Pipes
	QUERY PROCESSOR
	Plan Generation, Statistics, Cost Evaluation, Memory Grants, Parallel Operations
<b>STORAGE ENGINE</b>	
Access Methods, Locking, Transactions	
<b>SQL OS</b>	
Memory Management, Buffer Pool, Schedulers, Deadlock Detection, XEvents, Async I/O	

### 25.4.1 SQL Server Protocols

Elke applicatie die een connectie nodig heeft met SQL Server moet over het netwerk of lokaal op dezelfde server communiceren via een protocol layer. De opties hiervoor kunnen geconfigureerd worden in de **SQL Server Communication Manager -> SQL Server Network Configuration**:



In de meeste gevallen hebben we een goede configuratie van de TCP/IP instellingen nodig afhankelijk van de beschikbare poortnummers (default **TCP/1433**).

### 25.4.2 Query Processor

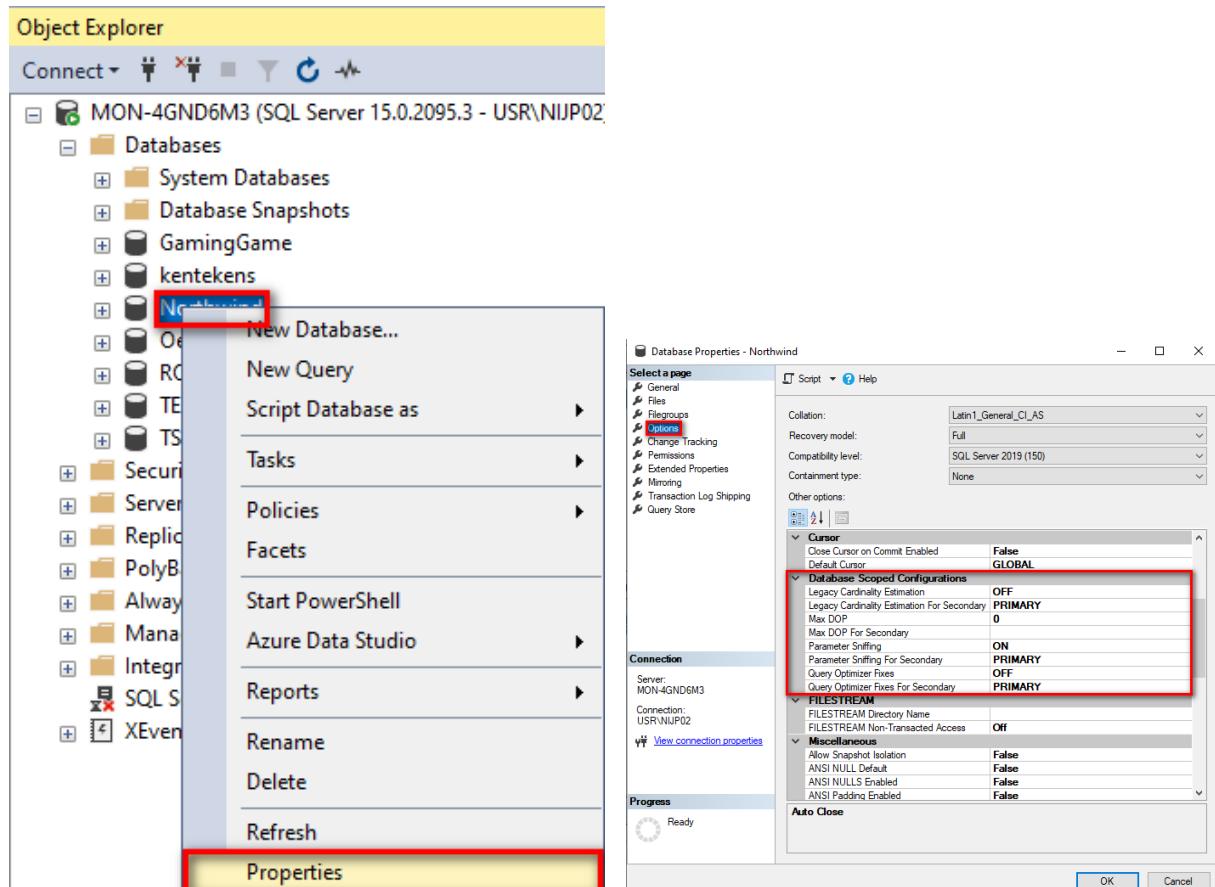
De **query processor** is een cruciaal deel van de architectuur en bevat meerdere interne componenten. We kunnen ze splitsen in **query optimization** voor het optimaliseren van de SQL statements en **query execution** voor het uitvoeren van het door de query optimizer berekende plan.

De **Query optimization** berekent de mogelijke wegen voor optimalisatie van het SQL statement en is gebaseerd op:

- **Cost**  
De cost geeft een indicatie van de kosten (geschatte CPU en I/O) van het plan
- **Cardinality**  
De cardinality geeft een indicatie van het aantal rijen dat verwerkt wordt (m.b.v. index en kolomstatistieken)

De cardinaliteit is voornamelijk gebaseerd op de WHERE clause van de query en SQL Server probeert een schatting te maken hoeveel records er verwerkt zullen worden door de query. Voor deze schatting worden index en kolom statistieken gebruikt. Hoe beter de schatting van de cardinaliteit is, des te beter de totale kosten van het plan.

We kunnen voor de configuratie van de query optimization de **Database Scope Configuration** gebruiken met <databasenaam> -> Properties -> Options:

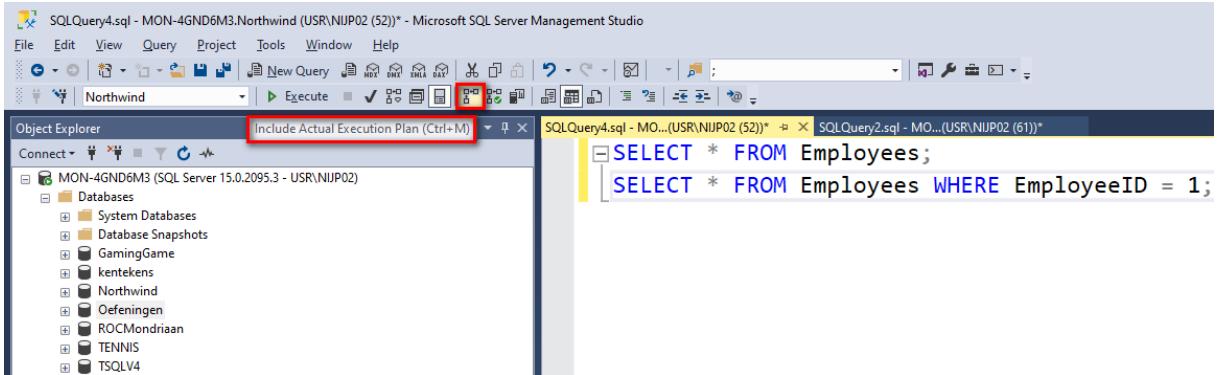


Onder Options zijn twee opties gerelateerd aan Cardinality Estimation, twee opties voor de maximale DPO (Degree of Parallelism), twee opties voor Parameter Sniffing (review van plan voor elke executie) en twee opties voor Query Optimizer Fixes (automatisch Query Optimizer updates installeren).

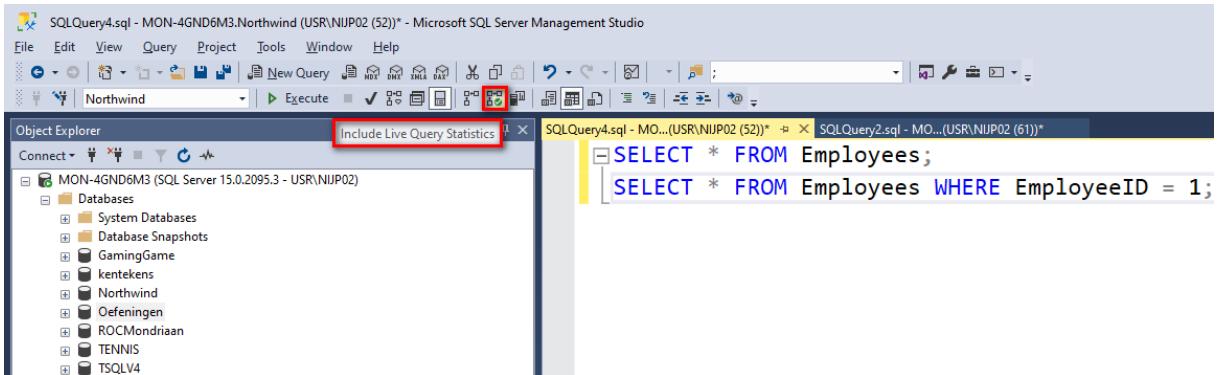
**Query execution** gebruikt het plan dat berekend is door de Query Optimizer en voert deze uit. Het uitvoeren van de query kan veel activiteiten op de storage engine en in memory veroorzaken om de nodige dataset terug te geven.

Om het execution plan in SSMS te zien kan je de volgende opties gebruiken:

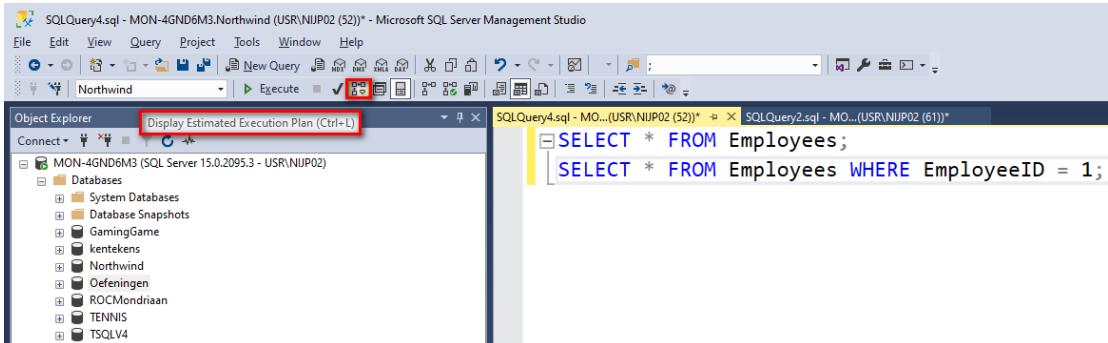
- **Include Actual Execution Plan (Control-M)**



- **Include Live Query Statistics**



- **Display Estimated Execution Plan (Control-L)**



Het estimated plan is berekend zonder het uit te voeren, terwijl de actual execution plan wel gebruikt is om de query activiteiten uit te voeren. Beide waarden kunnen van elkaar verschillen.

Wanneer we de data uit de tabellen moeten lezen, dan kunnen we de volgende algemene operatoren zien:

- **Table Scan**

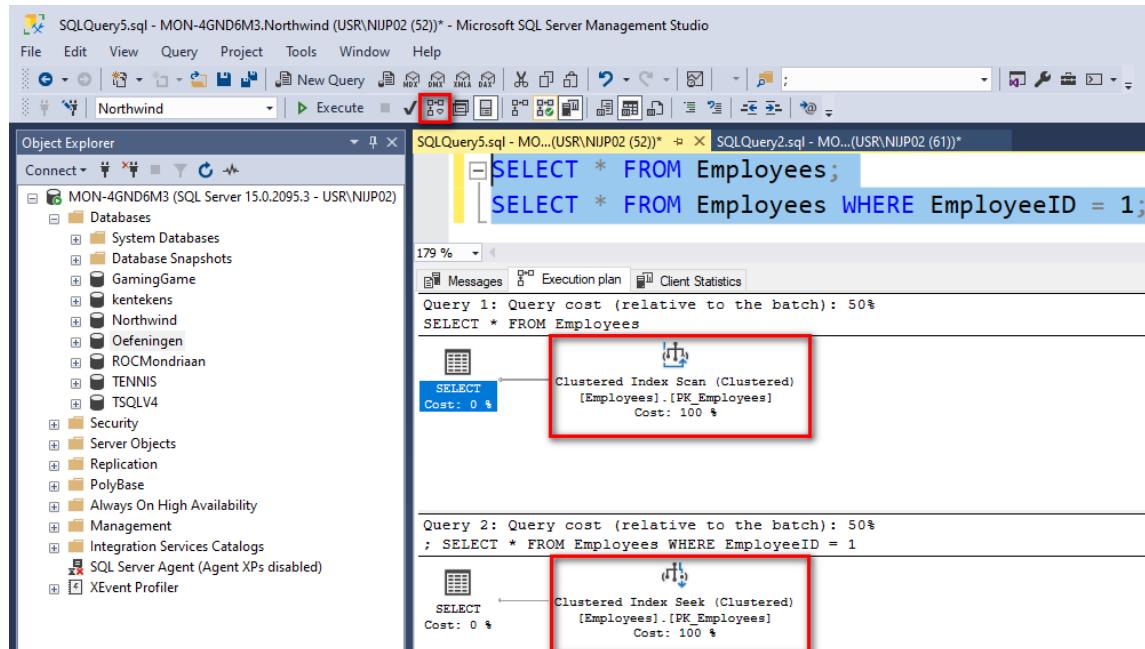
Dit haalt alle rijen van een tabel op die geen clustered index heeft (heap structure)

- **Clustered Index Scan**

Dit haalt alle rijen op van een tabel die een clustered index heeft

- **Columnstore index scan**  
Dit haalt alle rijen op van de columnstore index
- **Clustered Index Seek**  
Dit haalt alleen de rijen op gebaseerd op het zoek predicaat van de clustered index
- **Non-clustered index seek**  
Dit haalt alleen de rijen op gebaseerd op het zoek predikaat van de non-clustered index

Wanneer we queries opvatten dan zien we bij voorbeeld de volgende execution plans (uitvoeringsplannen):



In het voorbeeld zien we dat de query `SELECT * FROM Employees;` een Clustered Index Scan als executieplan heeft en dat `SELECT * FROM Employees WHERE EmployeeID = 1;` een Clustered Index Seek als executieplan heeft.

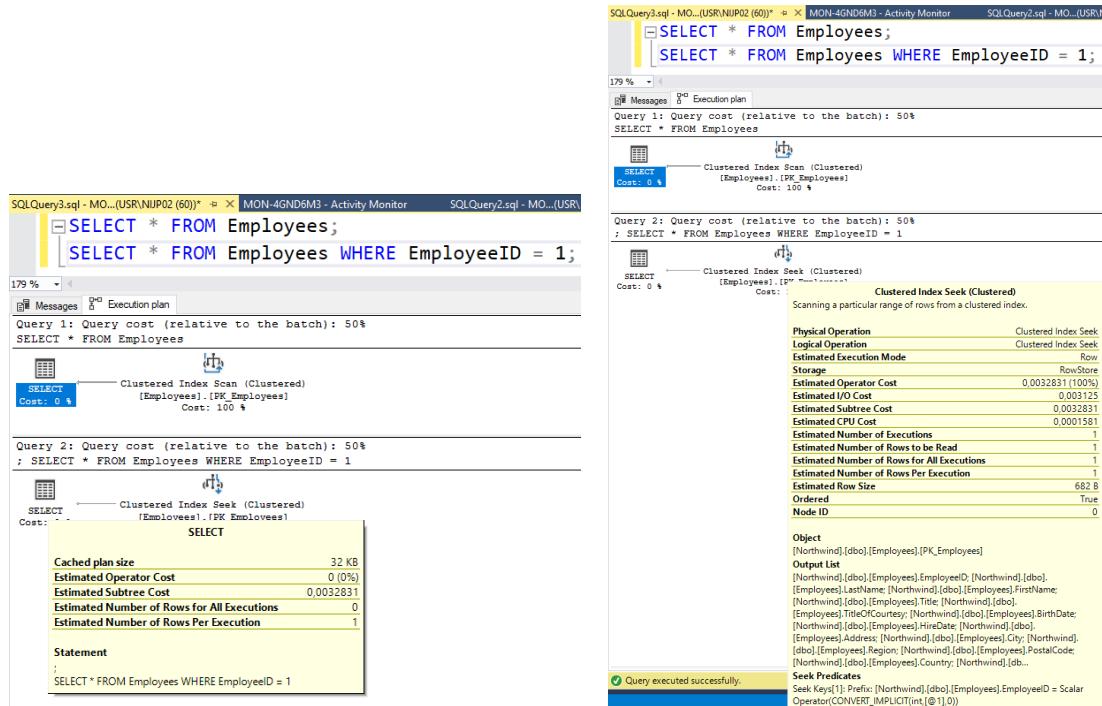
Een Index Scan haalt alle rijen op uit de tabel en een Index Seek haalt selectieve rijen op uit de tabel.

Aangezien een Index Scan elke rij in de tabel raakt (of deze nu in aanmerking komt of niet) zijn de kosten evenredig met het totale aantal rijen in de tabel. Een scan is dus een efficiënte strategie als de tabel klein is of als de meeste rijen (50% a 90%) in aanmerking komen voor het predicaat. Als er geen index is, zien we mogelijk een Table Scan (Index Scan) in het uitvoeringsplan.

Aangezien een Seek alleen betrekking heeft op rijen die in aanmerking komen en pagina's die deze in aanmerking komende rijen bevatten, zijn de kosten evenredig aan het aantal in aanmerking komende rijen en pagina's in plaats van aan het totale aantal rijen in de tabel. Index Seek hebben over het algemeen de voorkeur voor de zeer selectieve zoekopdrachten. Wat dat betekent is dat de query slechts een kleiner aantal rijen (minder dan 10 a 15 procent) opvraagt.

In het algemeen probeert de query optimizer een Index Seek te gebruiken, wat betekent dat de optimizer een bruikbare index heeft gevonden om recordset op te halen. Maar als het dit niet kan omdat er geen index of geen bruikbare indexen op de tabel staan, dan moet SQL Server alle records scannen die aan de queryvoorwaarde voldoen.

We kunnen nu de muis over operator bewegen zodat we details van de operator te zien krijgen:



```

SQLQuery3.sql - MO... (USR\NJP02 (60)) *  MON-4GND6M3 - Activity Monitor SQLQuery2.sql - MO... (USR)
SELECT * FROM Employees;
SELECT * FROM Employees WHERE EmployeeID = 1;

SQLQuery3.sql - MO... (USR\NJP02 (60)) *  MON-4GND6M3 - Activity Monitor SQLQuery2.sql - MO... (USR)
SELECT * FROM Employees;
SELECT * FROM Employees WHERE EmployeeID = 1;

179 % 179 %

Messages Execution plan
Query 1: Query cost (relative to the batch): 50%
SELECT * FROM Employees
Clustered Index Scan (Clustered)
[Employees].[PK_Employees]
Cost: 0 $ Cost: 100 $

Query 2: Query cost (relative to the batch): 50%
; SELECT * FROM Employees WHERE EmployeeID = 1
Clustered Index Seek (Clustered)
[Employees].[PK_Employees]
Cost: 0 $ Cost: 100 $

Scanning a particular range of rows from a clustered index.

Physical Operation Clustered Index Seek
Logical Operation Clustered Index Seek
Estimated Execution Mode Row
Storage RowStore
Estimated Operator Cost 0.0032831 (100%)
Estimated I/O Cost 0.003125
Estimated Subtree Cost 0.0032831
Estimated CPU Cost 0.0001581
Estimated Number of Executions 1
Estimated Number of Rows to be Read 1
Estimated Number of Rows for All Executions 1
Estimated Number of Rows Per Execution 1
Estimated Row Size 682 B
Ordered True
Node ID 0

Object [Northwind].[dbo].[Employees].[PK_Employees]
Output List [Northwind].[dbo].[Employees].[EmployeeID] [Northwind].[dbo].[Employees].[FirstName]
[Northwind].[dbo].[Employees].[LastName] [Northwind].[dbo].[Employees].[Title];
[Northwind].[dbo].[Employees].[TitleOfCourt] [Northwind].[dbo].[Employees].[BirthDate];
[Northwind].[dbo].[Employees].[HireDate] [Northwind].[dbo].[Employees].[City];
[Northwind].[dbo].[Employees].[Address] [Northwind].[dbo].[Employees].[Region];
[Northwind].[dbo].[Employees].[PostalCode] [Northwind].[dbo].[Employees].[Country];
[Northwind].[dbo].[Employees].[Gender];
Seek Predicates Seek Key(s): Prefix: [Northwind].[dbo].[Employees].[EmployeeID] = Scalar
Operator(COVERT_IMPLICIT(int,@1,0))

```

De belangrijke attributen om naar te kijken zijn:

- Het actuele aantal rijen versus het berekende aantal rijen
- Het aantal keren dat het uitgevoerd is
- Of er parallelisme is gebruikt
- Of er warnings zijn getoond

Operatoren die we kunnen zien bij nested lookup join zijn:

- Key lookup**  
Dit is een lookup op een tabel met een clustered index
- RID lookup**  
Dit is een lookup op een heap

Operatoren die we kunnen zien bij join operatoren zijn:

- Nested loop**  
Deze join wordt gebruikt voor veel join operaties wanneer er een rij in de outer tabel kijkt naar een rij in de inner tabel en deze terug geeft. Deze operator wordt gebruikt als de outer input veel rijen bevat en de lookup tabel klein is.
- Merge join**  
Dit algoritme is de meest efficiënte manier voor het joinen van twee grote sets gegevens die beiden gesorteerd zijn op de join key.

- **Hash match**

Dit type operator wordt gebruikt als het scenario niet past in een van de andere typen joins. Hash match wordt vaak gebruikt wanneer de tabellen niet goed gesorteerd zijn of als er geen indexen zijn.

Het is mogelijk om het execution plan in XML formaat op te slaan om aan anderen te zenden.

#### 25.4.3 Storage Engine layer

De Storage Engine layer is verantwoordelijk voor het benaderen van de data in de database en heeft twee hoofd componenten:

- Access Methods (het halen van de gegevens uit de data en index pages)
- Transaction and Locking

De user data is opgeslagen in data pages, row-overflow pages en LOB pages (voor Large objects). Als er een indexen zijn gebruikt worden die opgeslagen in index pages en ok row-overflow pages en LOB pages.

SQL Server gebruikt verschillende allocatie mappen voor het bijhouden van used/free space in de datafiles:

- **Page Free Space (PFS)**  
Deze pagina houdt een bitmap van het gebruik van de pages in de datafile
- **Global Allocation Map (GAM)**  
Deze houdt de extents bij die geallocaard zijn voor uniforme extents
- **Shared Global Area Map (SGAM)**  
Deze houdt de extents bij die als mixed extents zijn gebruikt

Een space (ruimte) in de database wordt beheerd door objecten die extents worden genoemd. Elke extent is gemaakt van 8 aaneengesloten pages. Uniforme extents worden gebruikt door één enkel object, dus alle 8 de pages van het extent behoren tot hetzelfde object. De mixed extents kunnen gedeeld worden door verschillende objecten.

#### 25.4.4 Performance Monitoring overview

Onder performance wordt verstaan de responsetijd van een verzoek. Een slechte performance betekent dat de responsetijd voor de client (afnemer) onacceptabel is. Een slechte performance kan veroorzaakt worden door veel factoren, zoals een slechte query syntax, ontbrekende indexen of een slechte netwerkconfiguratie.

Bij performanceproblemen moeten we de root cause (oorzaak) die de slechte performance veroorzaakt vinden. Op basis van voorgaande ervaringen is het goed om de volgende punten te beschouwen:

- **Number of requests (aantal verzoeken)**  
Een benchmark (vergelijkend onderzoek waarbij de prestaties worden onderzocht) waarde die de invloed weergeeft van de hoeveelheid gelijktijdige acties op gegevens en CPUs gebruikt door de requests.
- **Space effected by the request (plaats die gevolgen heeft door het verzoeken)**  
Hoeveel gegevens zijn er verplaatst tussen de harde schijf en het geheugen? Is deze

hoeveelheid gegevens noodzakelijk voor het vervullen van de verzoeken of kan het worden verminderd?

- **Request type (typen verzoeken)**

Wordt de SQL Server instance gevraagd door veel kleine random I/O acties of door grote scannende acties zoals aggregatie queries?

- **Request difficulty (moeilijkheid van de verzoeken)**

Zijn de requests eenvoudig (alleen gegevens van één tabel lezen) of zijn ze complex (veel JOIN operaties, gecompliceerde condities, etc.)?

Daarnaast dienen we te weten of andere gelijktijdige services, zoals reporting service of antivirusprogramma's dezelfde resources delen met de SQL Server instance.

Wanneer we de performance monitoren en troubleshooten dan hebben we twee perspectieven:

- **Performance baseline (proactieve)**

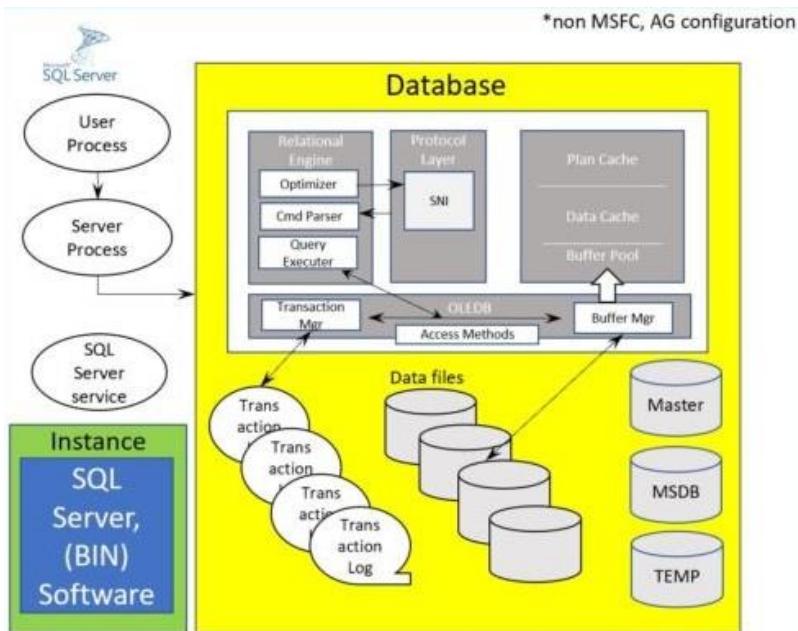
Weten hoe het normale gedrag van het OS en SQL Server is, zodat we afwijkingen kunnen vaststellen en de reden ervan kunnen signaleren.

Een performance baseline is een set maatstaven van het systeem die het normale performance gedrag laat zien, zoals CPU, geheugen, maar ook de wijzigingen hierop in de loop van de tijd.

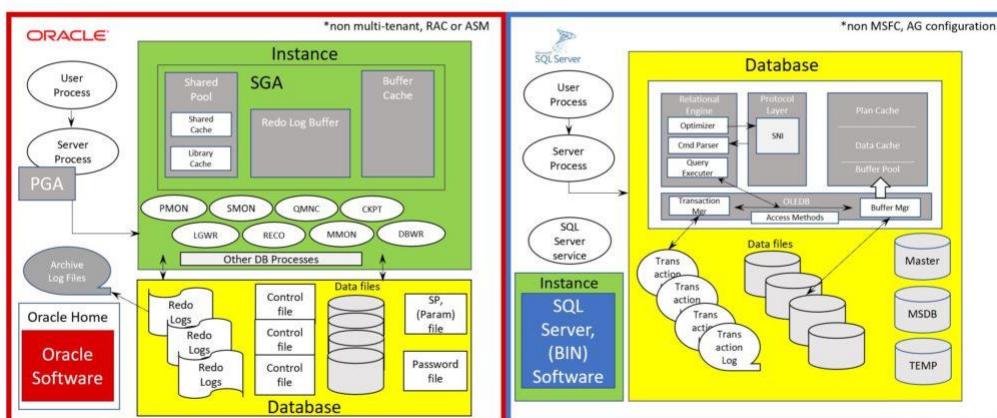
Met de performance baseline kunnen we performanceproblemen voorkomen voordat ze zich voordoen.

We kunnen hiervoor onze eigen tools gebruiken, zoals Performance Monitor en Extended Events, maar een goede effectieve tool is Data Collection.

- **Reageren op situaties waarbij SQL Server stopt met het response in een betekenisvolle weg.**  
De databasebeheerder monitort top-down de performance, te beginnen bij de instance. Een instance is een kopie van de executable `sqlservr.exe` die draait als een operation system service. Elke instance beheert een aantal system databases en één of meerdere user databases. Op een computer kunnen meerdere instances van de database engine draaien. Applicaties maken verbinding met de instance om werkzaamheden op de database uit te voeren die beheerd wordt door de instance.

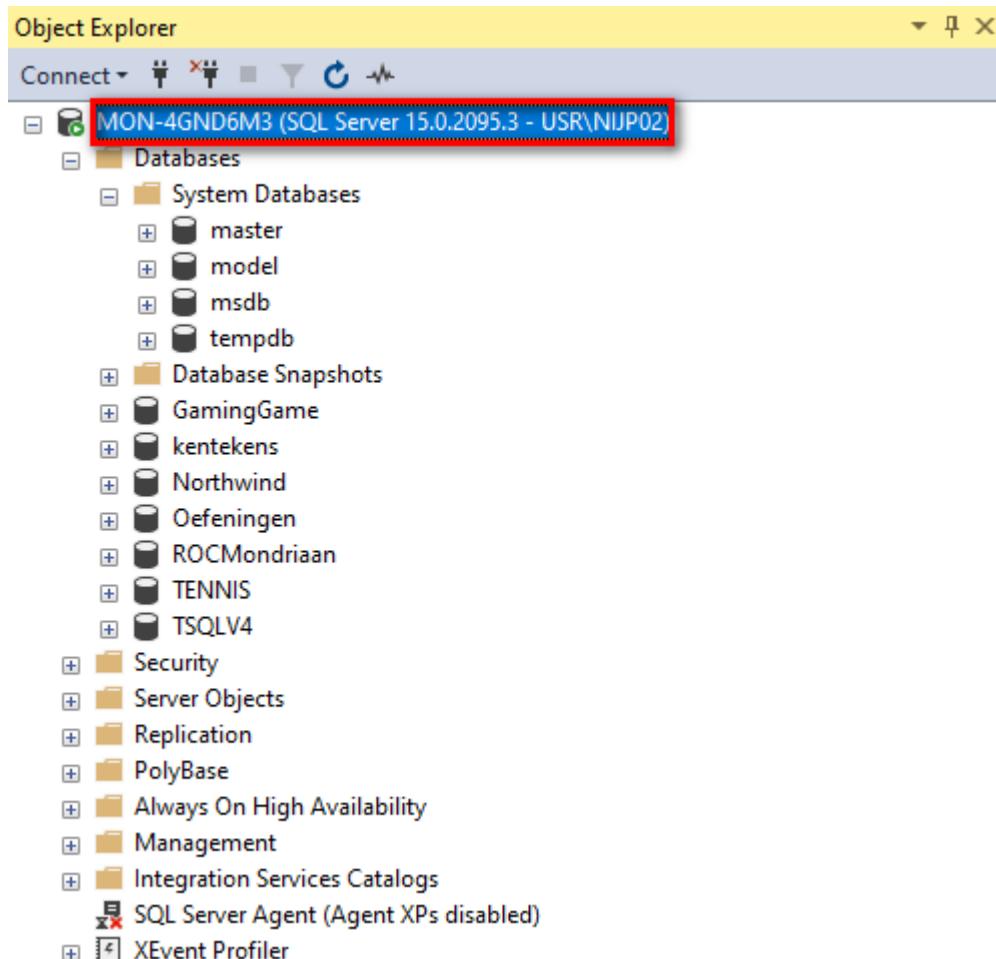


## Oracle Compared to SQL Server, (High Level)



In SSMS is een instance het hoofste niveau in de boom, met daaronder een aantal system

databases en één of meerdere user databases:

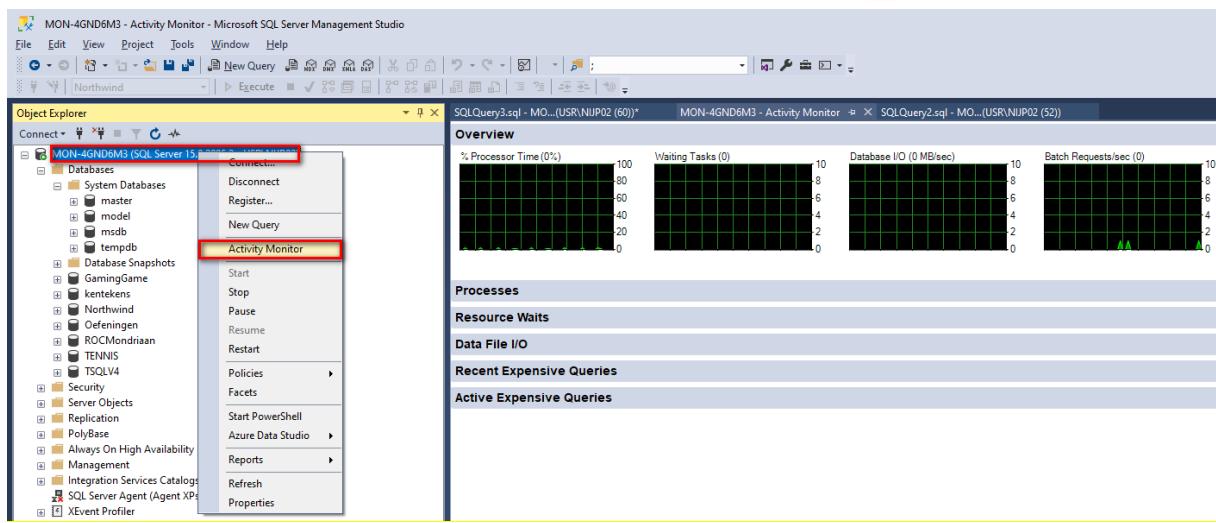


#### 25.4.5 Tools voor het monitoren van de performance

Om de performance te kunnen tunen moeten we weten welke tool we kunnen gebruiken en hoe we de resultaten moeten interpreteren.

#### 25.4.6 Activity Monitor

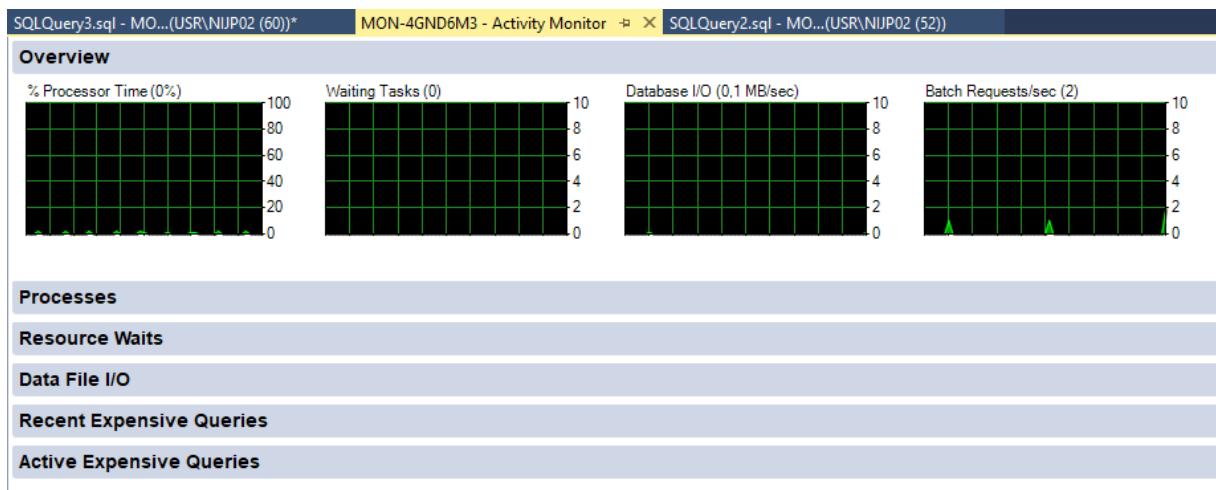
De Activity Monitor is een snelle en relatief eenvoudige tool binnen SSMS, die opgestart kan worden door rechtermuisklik op de instance -> Activity Monitor



De Activity Monitor is onderverdeeld in zes secties:

- Overview
- Processes
- Resource Waits
- Data File I/O
- Recent Expensive Queries
- Active Expensive Queries

**Overview** geeft snel informatie over de huidige CPU gebruik door SQL Server, het aantal taken die op een resource staan te wachten, totale hoeveelheid databewegingen tussen fysieke disk en de buffer cache en het huidige aantal requests. Default (standaard) wordt het scherm elke 10 seconden refreshed. Het geeft de huidige status van de instance, géén uitgebreide gedetailleerde gegevens.



**Processes** geeft een grid (rooster) met alle sessies die momenteel uitgevoerd zijn op de SQL Server Instance met de huidige status.

SQLQuery3.sql - MO... (USR\NJP02 (60))*											MON-4GND6M3 - Activity Monitor	SQLQuery2.sql - MO... (USR\NJP02 (52))
Overview												
Processes												
S.	U.	Login	Dat...	Task State	Com...	Application	Wait Tim...	Wait Type	Wait Resource	B...	H...	Me...
51	1	USR\NJP02	master			Microsoft SQL Server Management Studio	0					0 MON-4GND6M3
52	1	USR\NJP02	Northwind			Microsoft SQL Server Management Studio - Query	0					32 MON-4GND6M3
53	1	NT SERVICE\SQLTELEMETRY	master			SQLServerCEIP	0					32 MON-4GND6M3
57	1	USR\NJP02	tempdb	RUNNING	SELECT	Microsoft SQL Server Management Studio	0					32 MON-4GND6M3
58	1	USR\NJP02	master			Microsoft SQL Server Management Studio	0					32 MON-4GND6M3
60	1	USR\NJP02	Northwind			Microsoft SQL Server Management Studio - Query	0					32 MON-4GND6M3

Resource Waits

Data File I/O

Recent Expensive Queries

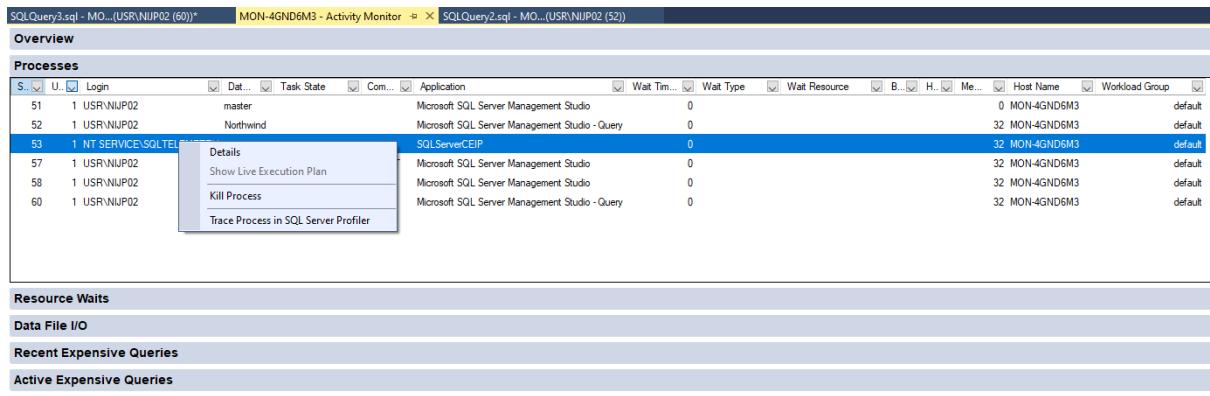
Active Expensive Queries

De kolommen van sectie Processes zijn:

- **Session ID** is de unieke identifier van de sessie (SPID)
- **User process flag** is standaard gefilterd op 1 (alleen user sessions). Door op de kolomnaam te klikken kan het veranderd worden naar alle sessies.
- **Login name** is de login context van de gegeven sessie
- **Database** is de database context van de gegeven sessie
- **Task state** kan zijn:
  - Empty** betekent dat de sessie bestaat maar geen actuele request heeft voor de instance
  - RUNNING** betekent dat de request op dit moment wordt uitgevoerd door SQL Server
  - SUSPENDED** betekent dat de taak staat te wachten op een resource die geblockt is door een andere sessie
- **Current type of command** is niet het commando zelf maar is het type commando (SELECT, ROLLBACK,...)
- **Application** is de naam van de applicatie die optioneel geschreven is de connection string
- **Wait time** is de tijd dat de sessie staat te wachten op een blocked resource
- **Wait type** is het type lock die door een andere sessie wordt gehouden en de sessie staat te wachten
- **Wait resource** is een beschrijving van het object waarop een incompatible lock is gezet door een andere sessie
- **Blocked by** is een van de meest belangrijke kolommen en geeft aan welke sessie de block voor de betreffende sessie heeft gezet.
- **Head Blocker** bevat de vlag 1 wanneer de sessie resources lockt zonder dat het staat te wachten op andere blockers en dus zelf de blocker is. We kunnen het blocking conflict oplossen door deze sessie te killen.
- **Memory usage** geeft aan hoeveel geheugen (in kB) de sessie op dit moment heeft geconsumeerd
- **Host name** is de naam van de SQL Server instance
- **Workload group** is de resource group waar de sessie aan toegekend is. In SQL Server kan m.b.v. de Resource Governor workload groups worden gebruikt voor een fair usage policy (eerlijk gebruik beleid).

Wanneer we met de muis over de kolomnamen bewegen dan komt er een tooltip over de source DMV van de kolom.

Wanneer we met de rechtermuis op een proces kliken dan komt er een pop-up menu.



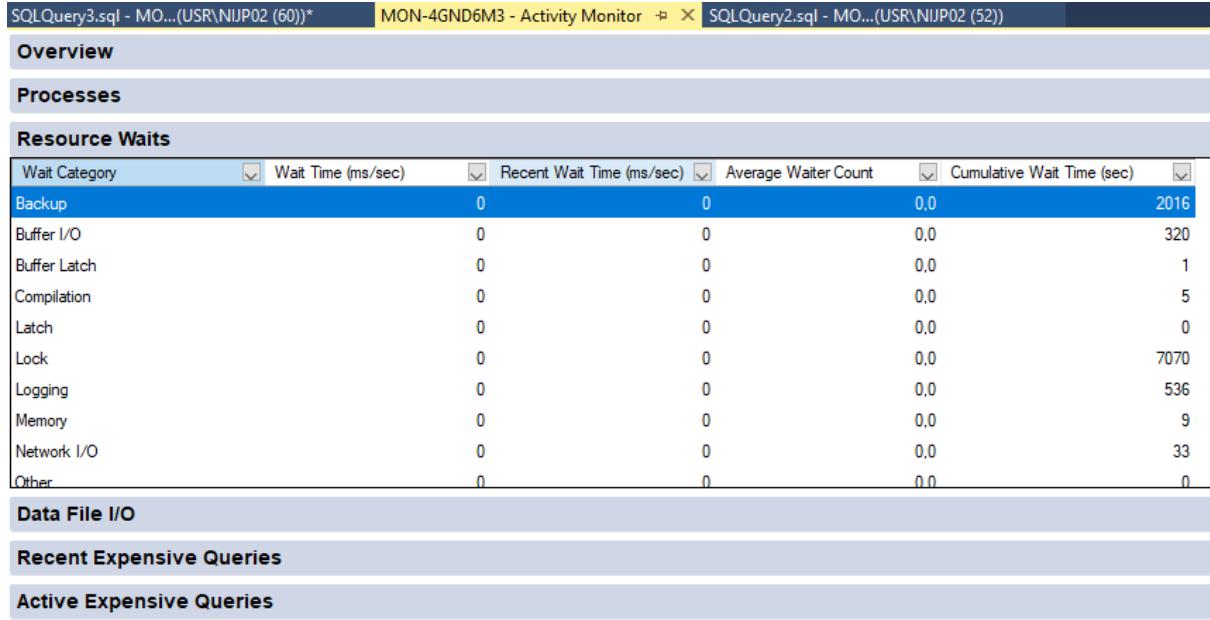
The screenshot shows the SQL Server Activity Monitor interface. In the center, there is a list of processes. One specific process, identified by ID 53 and labeled 'NT SERVICE\SQLTELE', has a context menu open over it. The menu contains four items: 'Details', 'Show Live Execution Plan', 'Kill Process', and 'Trace Process in SQL Server Profiler'. The rest of the interface includes tabs for Overview, Processes, Resource Waits, Data File I/O, Recent Expensive Queries, and Active Expensive Queries.

De opties in het pop-up menu zijn:

- **Details** geeft de tekst van de laatst uitgevoerde of op dit moment uitgevoerde commando
- **Show Live Execution Plan** geeft het execution plan van de op dit moment uitgevoerde query
- **Kill Process** is een optie om een sessie killen, bijvoorbeeld wanneer het een head blocker is
- **Trace Process in SQL Server Profiler** is een optie om de **SQL Server Profiler** op te starten die elke activiteit van de sessie speurt

SQL Server lost geen unacceptable blocking waits op tussen sessies. Het kill statement moet handmatig worden uitgevoerd door de databasebeheerder. Kijk altijd eerst naar de process details voordat je een proces killt. Je wilt niet degene zijn die een bedrijf kritische taak heeft gekilled!

**Resource waits** geeft een overzicht waar sessies op wachten.



The screenshot shows the SQL Server Activity Monitor with the 'Resource Waits' tab selected. A table is displayed, listing various wait categories along with their wait times and counts. The columns are: Wait Category, Wait Time (ms/sec), Recent Wait Time (ms/sec), Average Waiter Count, and Cumulative Wait Time (sec). The data shows that 'Backup' is the most frequent wait type, followed by 'Lock' and 'Network I/O'. Other categories like 'Buffer I/O' and 'Memory' also appear in the list.

Wait Category	Wait Time (ms/sec)	Recent Wait Time (ms/sec)	Average Waiter Count	Cumulative Wait Time (sec)
Backup	0	0	0,0	2016
Buffer I/O	0	0	0,0	320
Buffer Latch	0	0	0,0	1
Compilation	0	0	0,0	5
Latch	0	0	0,0	0
Lock	0	0	0,0	7070
Logging	0	0	0,0	536
Memory	0	0	0,0	9
Network I/O	0	0	0,0	33
Other	0	0	0,0	0

**Data File I/O**

**Recent Expensive Queries**

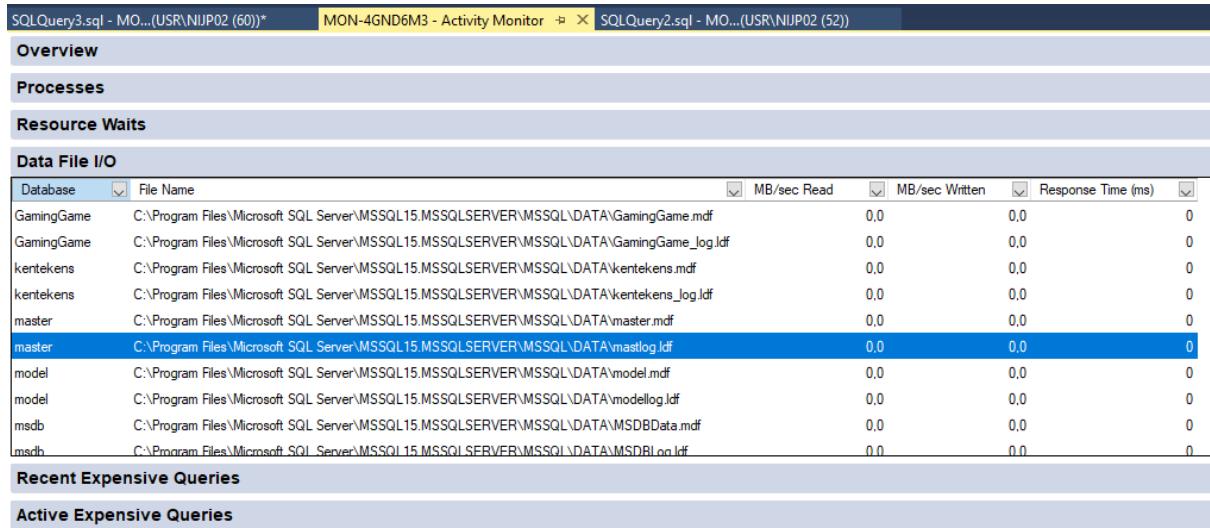
**Active Expensive Queries**

De kolommen van de Resource Waits spreken voor zich. Alleen de eerste kolom Wait Category heeft extra uitleg nodig. SQL Server herkent honderden wait types, en ze zijn niet allemaal goed gedocumenteerd. Wij dienen slechts enkele basis categorieën te kennen die alledaags zijn voor de werking van SQL Server:

- **Page Latch** betekent dat er latches (lichtgewicht locks) zijn op pages in de buffer cache. SQL Server gebruikt dit om data pages gedurende lees en schrijf acties te beschermen. Oorzaak kan zijn dat er te veel indexen zijn op de tabel of door verwaarloosd index onderhoud.
- **Page IO Latch** betekent dat er korte page locks zijn voor lezen en schrijven van of naar disk. Oorzaak kan zijn dat er te veel IO operaties zijn op een klein stukje geheugen of een trage disk subsystem.
- **Latch** betekent dat er latches (korte lichtgewicht locks) zijn op andere resources. Een te groot aantal latches maakt diepgaandere diagnose noodzakelijk.
- **Lock** betekent dat er op delen van de tabel (rij, page, extent, etc.) zijn door veel gelijktijdige gebruikers, of een onjuist beheerde transactie of te veel indexen of te oude statistieken
- **CXPACKET** betekent dat er gewacht wordt op de dispatcher thread bij parallel taken en kan betekenen dat er parallel taken worden uitgevoerd op eventuele gedistribueerde gegevens
- **LOGBUFFER** betekent dat er gewacht wordt op het schrijven van transaction log records naar de buffer doordat er te veel kleine transacties op hetzelfde moment uitgevoerd worden

Niet elke wait is een probleem. Het is heel gebruikelijk dat unacceptable waits een symptoom zijn van een ander probleem.

**Data file I/O** geeft de huidige stroom van gegevens van elk bestand van elke database.

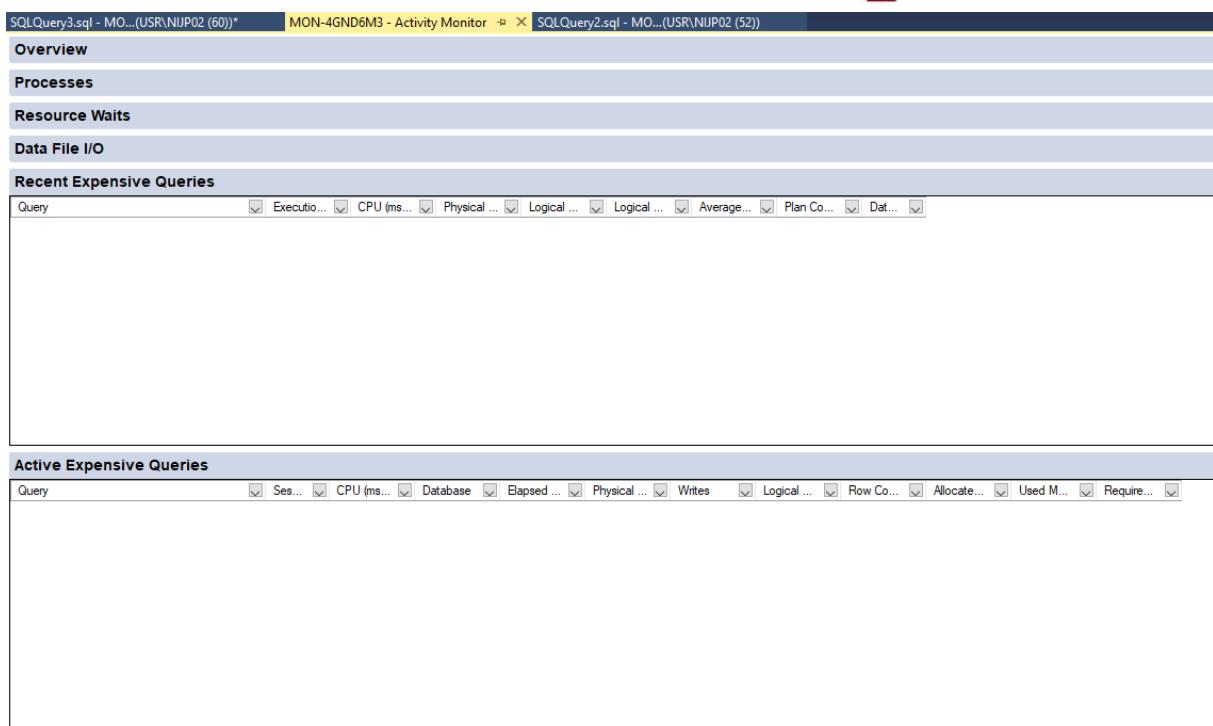


The screenshot shows the Activity Monitor interface with three tabs: Overview, Processes, and Resource Waits. The Resource Waits tab is selected, displaying a table titled 'Data File I/O'. The table lists various databases along with their file names, MB/sec Read, MB/sec Written, and Response Time (ms). The data is as follows:

Database	File Name	MB/sec Read	MB/sec Written	Response Time (ms)
GamingGame	C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\MSSQL\DATA\GamingGame.mdf	0,0	0,0	0
GamingGame	C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\MSSQL\DATA\GamingGame_log.ldf	0,0	0,0	0
kentekens	C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\MSSQL\DATA\kentekens.mdf	0,0	0,0	0
kentekens	C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\MSSQL\DATA\kentekens_log.ldf	0,0	0,0	0
master	C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\MSSQL\DATA\master.mdf	0,0	0,0	0
master	C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\MSSQL\DATA\mastlog.ldf	0,0	0,0	0
model	C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\MSSQL\DATA\model.mdf	0,0	0,0	0
model	C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\MSSQL\DATA\modellog.ldf	0,0	0,0	0
msdb	C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\MSSQL\DATA\MSDBData.mdf	0,0	0,0	0
msdb	C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\MSSQL\DATA\MSDBLog.ldf	0,0	0,0	0

Below the table, there are two sections: 'Recent Expensive Queries' and 'Active Expensive Queries'.

**Recent Expensive Queries** en **Active Expensive Queries** geven de meest dure queries met statistieke informatie. De recent expensive queries betreffen de queries die afgerond zijn terwijl de Active Expensive Queries de queries betreffen die op dit moment uitgevoerd worden.



The screenshot shows the MON-4GND6M3 - Activity Monitor window with two main sections:

- Recent Expensive Queries:** This section lists queries based on execution frequency. The columns shown in the header are: Query, Executio..., CPU (ms...), Physical ..., Logical ..., Logical ..., Average..., Plan Co..., Dat... . The table body is currently empty.
- Active Expensive Queries:** This section lists queries currently running. The columns shown in the header are: Query, Ses..., CPU (ms...), Database, Elapsed..., Physical..., Writes, Logical..., Row Co..., Allocate..., Used M..., Require... . The table body is currently empty.

De Recent Expensive Queries heeft de volgende kolommen:

- **Number of executions per minute** is het aantal keren dat de query per minuut was uitgevoerd
- **Number of milliseconds spend by the query on CPU per second** is de verhouding van tijd dat was besteed op CPU per minuut
- **Physical reads per second** is het aantal data pages dat van fysieke disk is gelezen
- **Logical Reads per second** is het aantal reads waarbij de data in de buffer cache aanwezig was
- **Logical Writes per second** is het aantal writes waarbij de data in de buffer cache aanwezig was
- **Average duration of the query** is de gemiddelde hoeveelheid tijd die nodig was voor het aantal executies
- **Plan count** is het aantal plan versies die in de procedure cache is geplaatst
- **Database name** is de database context van elke query

De Active Expensive Queries is gelijk aan die van de Recent Expensive Queries in de laatste 30 seconden en heeft de volgende extra kolommen:

- **Session ID** is de SPID van de aanvrager van de query
- **Elapsed time** is de totale nodige tijd voor het uitvoeren van alle acties van de query
- **Row count** is het aantal rijen dat door de query executie verwerkt is
- **Allocated Memory** is het toegekende geheugen dat is toegekend voor de query uitvoering
- **Requested memory** is de hoeveelheid aangevraagde geheugen allocatie. Indien dit later is dan de allocated memory dan kan dat inefficiënt geheugen betekenen.
- **Used Memory** is het geheugen dat door de query is gebruikt

In beide schermen is het mogelijk met een rechtermuisklik op een query een execution plan van de query te verkrijgen.

Alle gegevens in de Activity Monitor wordt gewist bij het opnieuw opstarten van de instance.

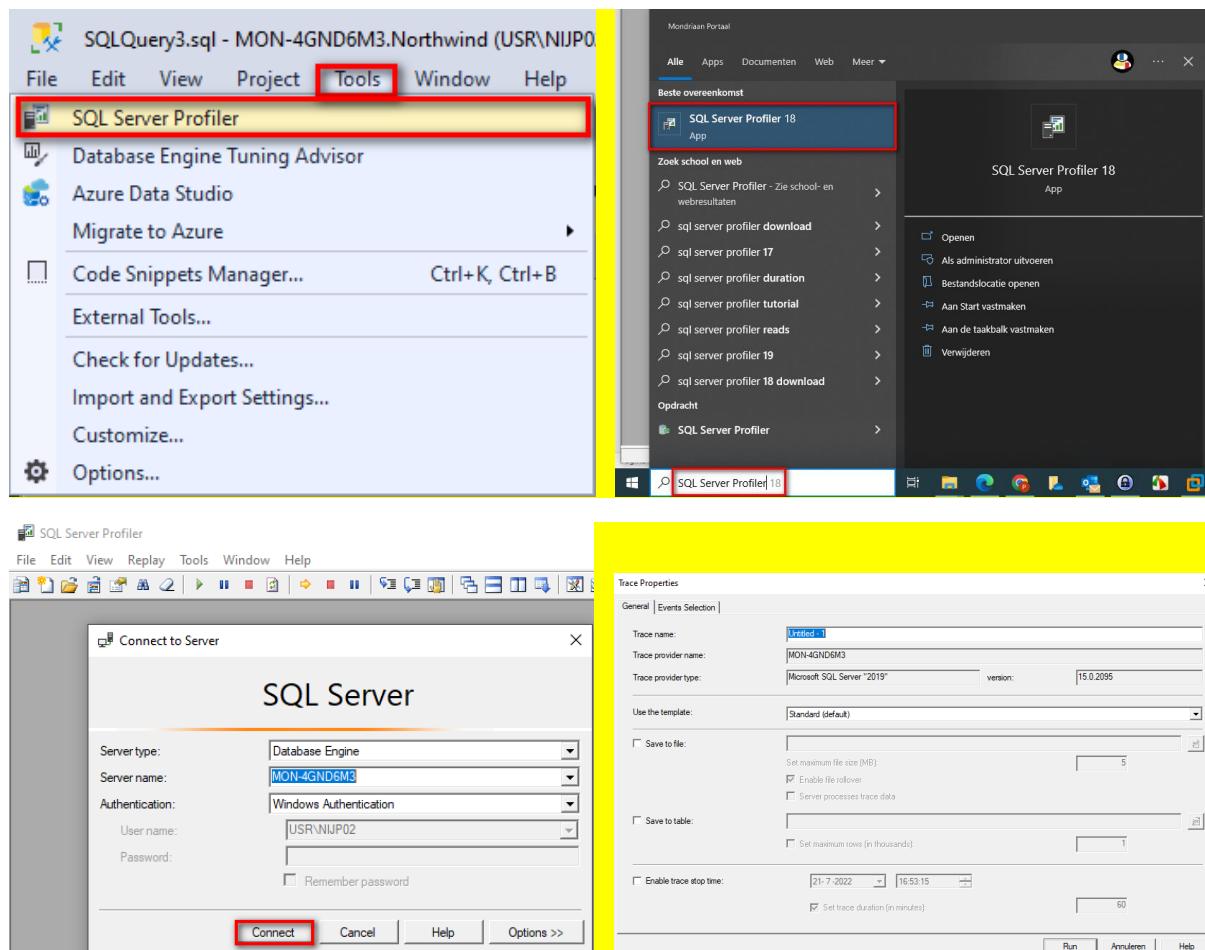
#### 25.4.7 Performance Monitor (TODO)

De performance monitor is een algemeen bekende tool binnen Windows OS en kan performance statistieken tonen van bijna elk deel van het OS en services die er op draaien. (zie hoofdstuk 2 Keeping your SQL server environment Healthy.)

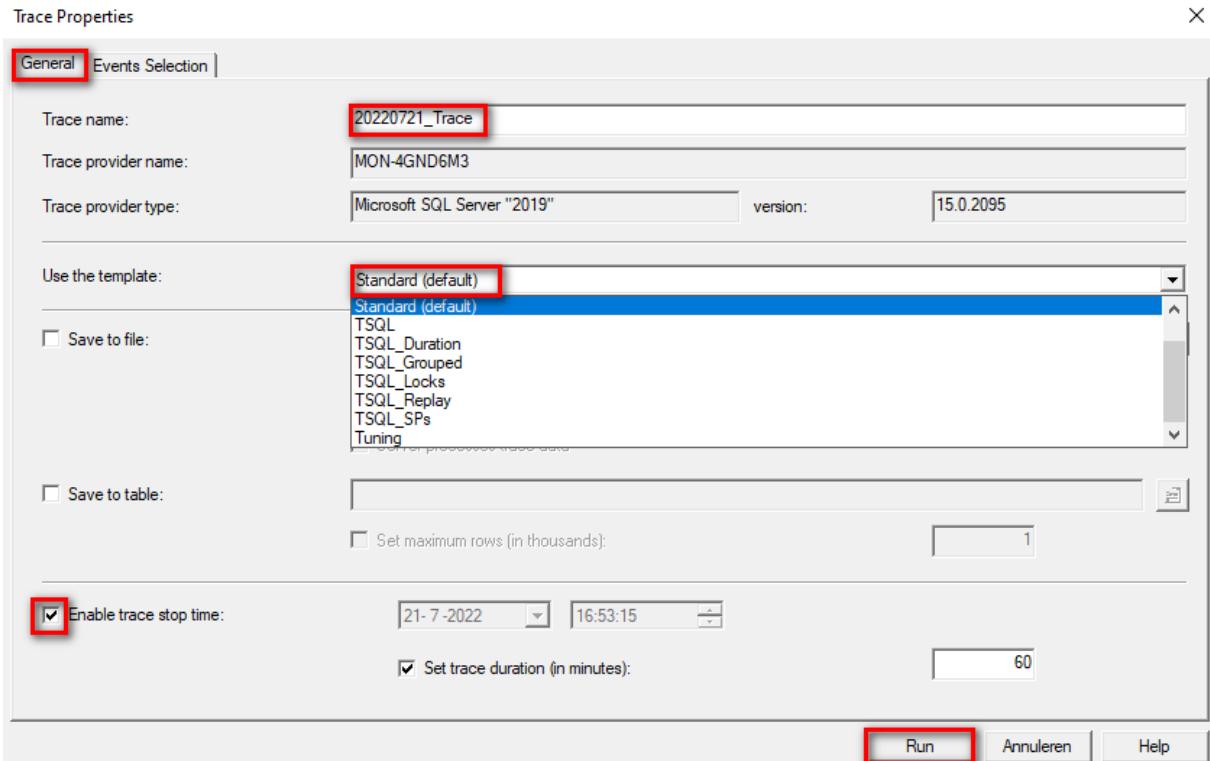
#### 25.4.8 SQL Server Profiler

SQL Server Profiler is een client tool die gebruikt kan worden om traces te creëren en uit te voeren. Traces zijn een set van events van een bepaald type waarin de databasebeheerder geïnteresseerd is. De traces kunnen bewaard worden in bestanden of in database tabellen.

SQL Server Profiler kan vanuit SSMS opgestart worden met Tools -> SQL Server Profiler -> Connect of vanuit het startmenu van Windows -> SQL Server Profiler 18 -> Microsoft SQL Server Management -> New Trace -> Connect:



Omdat er per SQL Server Profiler instance meerdere traces kunnen worden gemaakt, kan in het General tabblad aan elke trace een naam worden gegeven. SQL Server Profiler heeft een aantal voor gedefinieerde templates die eenvoudige keuzes van events mogelijk maken, maar we kunnen ook zelf templates definiëren.

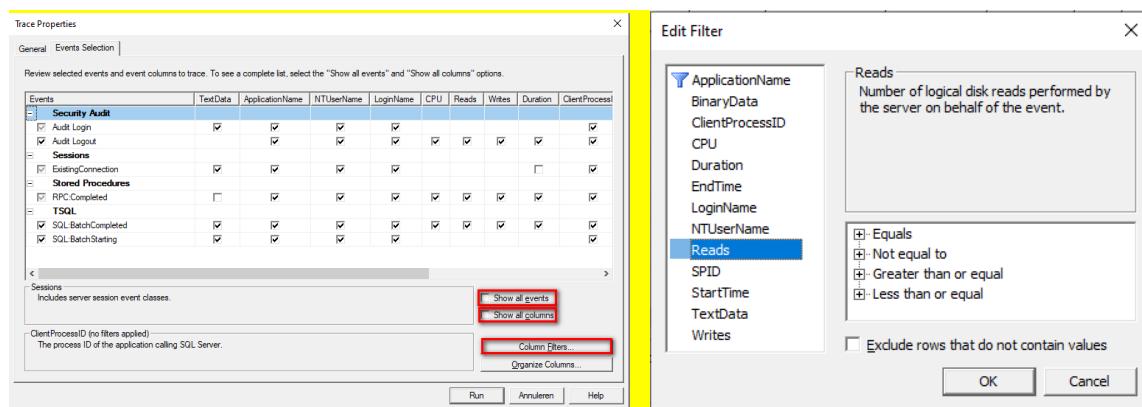


**Default wordt het resultaat niet bewaard, maar het is mogelijk om op te slaan naar een bestand of naar tabel in database met Save to file of met Save to table.**

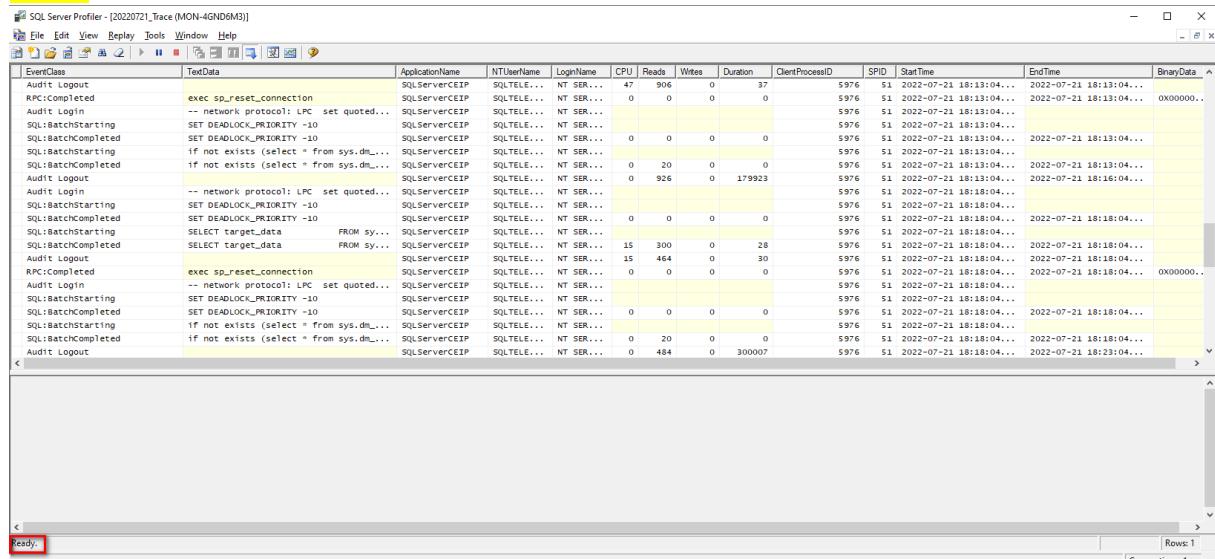
**De laatste optie Enable trace to stop time is erg nuttig omdat het vaak voorkomt dat een databasebeheerder SQL Server Profiler opstart en dan onbeheerd achter laat voor een lange tijd terwijl SQL Server Profiler veel resources verbruikt en daarmee de server zelfs kan doen crashen.**

**In het Events Selection tabblad kunnen we de gewenste events selecteren of deselecteren. Het is een goed ding om alleen de kolommen te selecteren die nodig zijn voor de diagnose. Zet daarom rechtsonderin Show all events en Show all columns uit als het niet nodig is.**

**Met Column Filters is het mogelijk om indien gewenst nog een filter op een kolom te zetten.**



Wanneer we op Run klikken dan sluit het dialoog scherm en voert SQL Server Profiler op de achtergrond enkele activiteiten uit, genereert een script en voert deze uit en stuurt de opgevangen data terug naar SQL Server Profiler en opent dan een Trace window om de events uit de events te tonen.



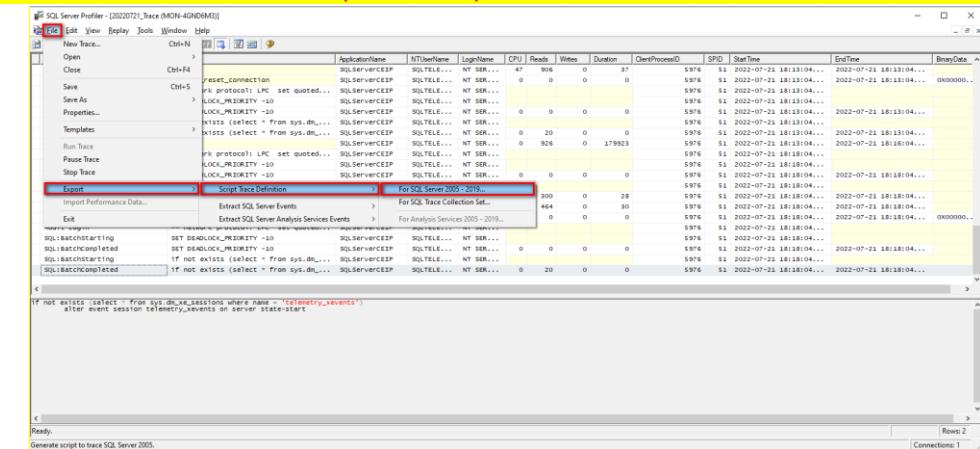
SQL Server Profiler is een bekende tool, maar heeft twee nadelen. De eerste is dat het veel resources gebruikt en de tweede is dat het door Microsoft als verouderd is gemarkerd en er sinds 2012 geen nieuwe features aan worden toegevoegd. Wanneer we inkomende events willen monitoren dan hoeven we niet telkens SQL Server Profiler op te starten maak kan wel helpen met het genereren van een script voor SQL Trace. Het script heeft een extensie .blk.

#### 25.4.9 SQL Trace

SQL Trace is een niet visuele server side tracing tool van SQL Server terwijl SQL Server Profiler een client-side viewer is. SQL Trace is gedefinieerd als een set Stored Procedures.

Het schrijven van een SQL Trace script is vrij moeilijk. Daarom is het handiger om een script te genereren in SQL Server Profiler:

1. Definieer en draai een nieuwe trace in SQL Server Profiler
2. Vanuit het File menu -> Export -> Script trace definition -> For SQL Server 2005 – 2019...

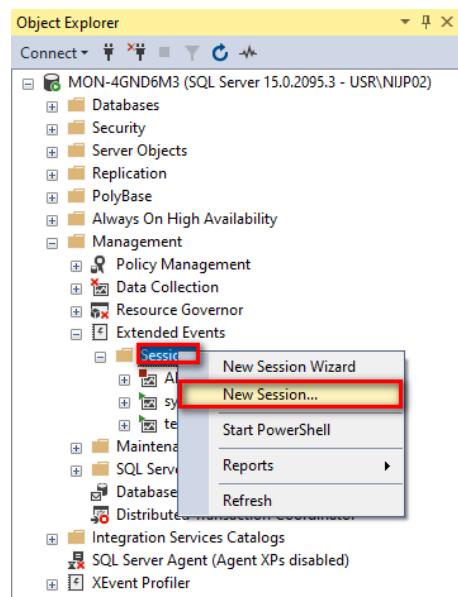


3. Save as om het script op te slaan

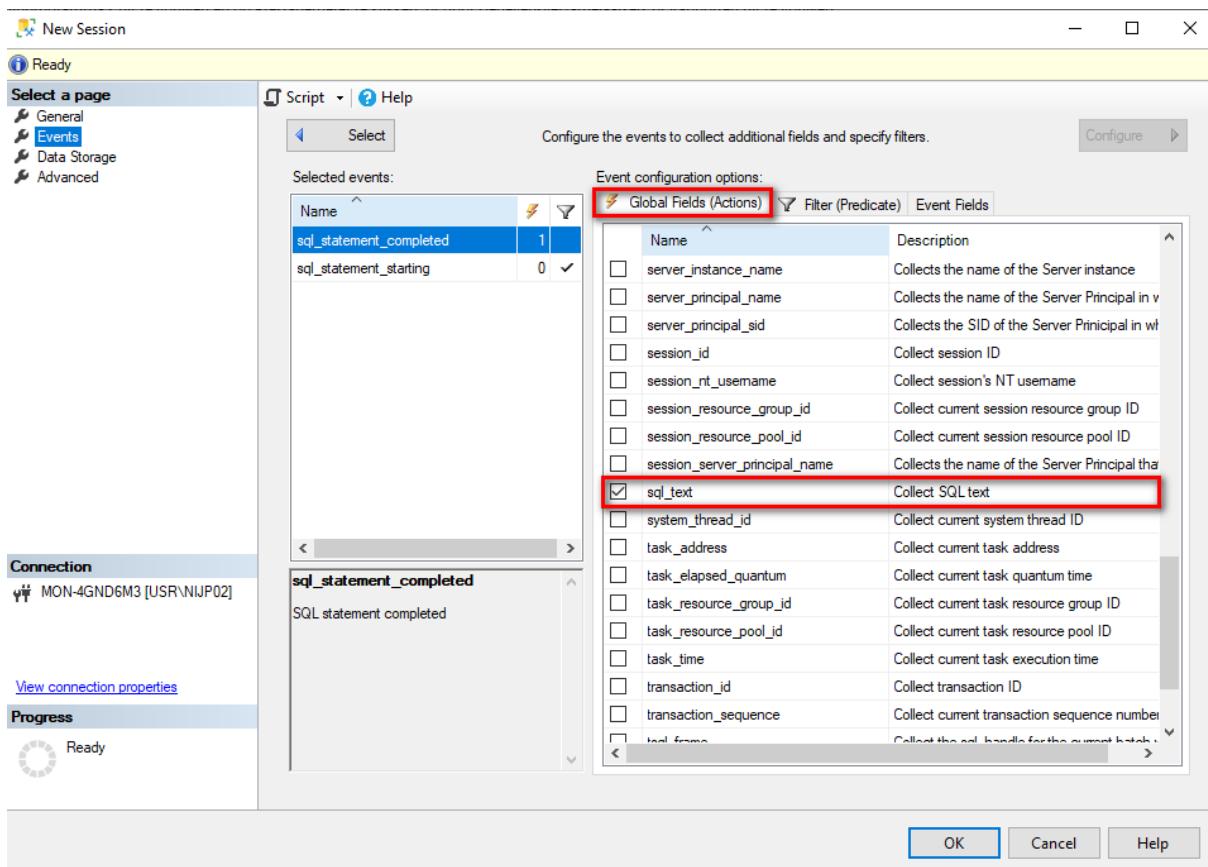
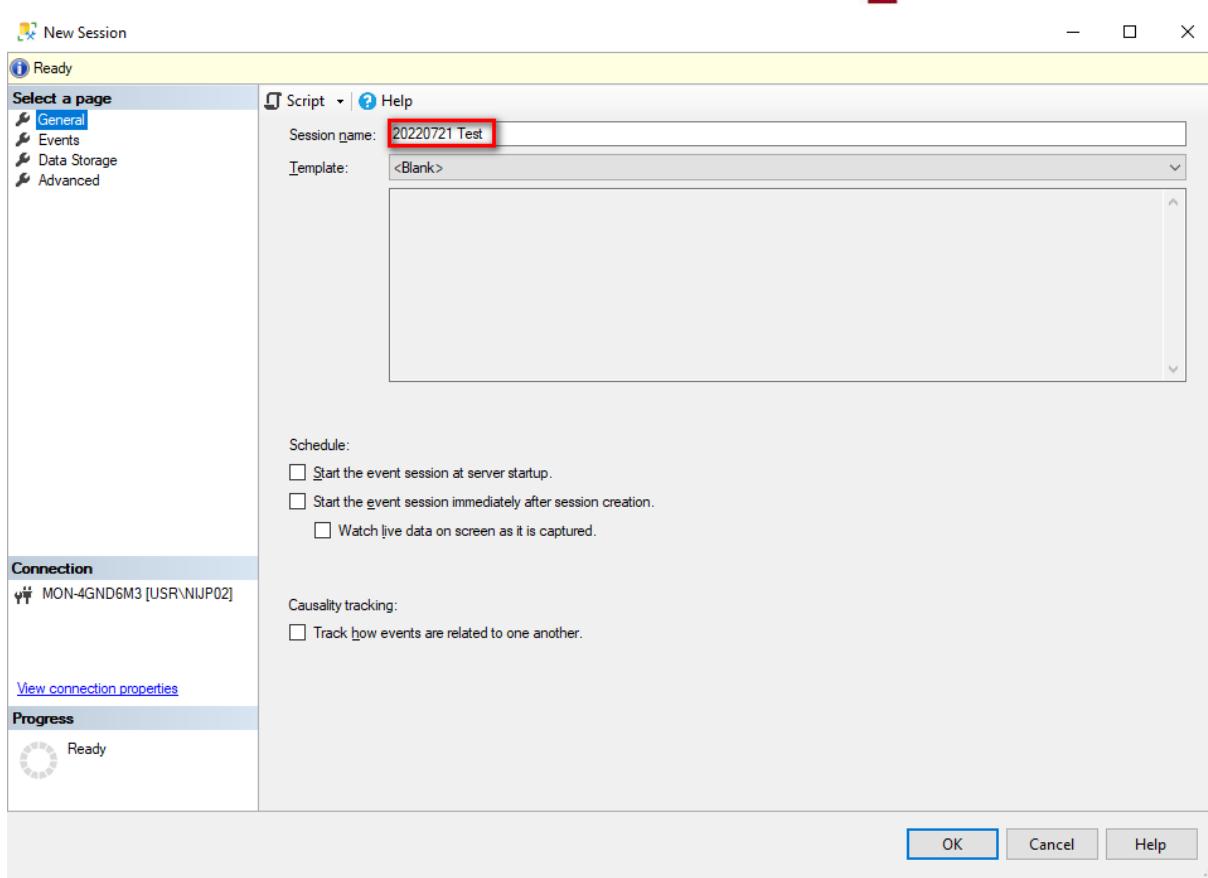
#### 25.4.10 Extended EventsXE

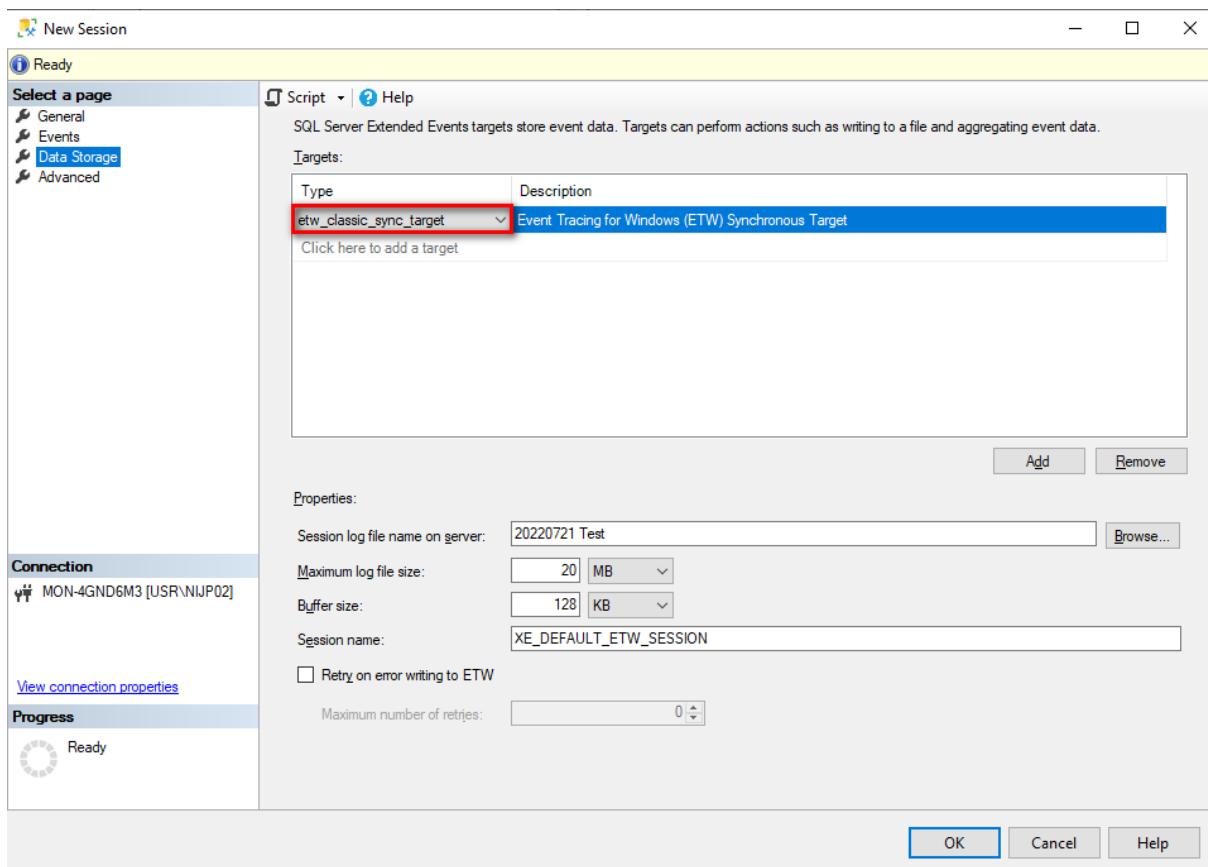
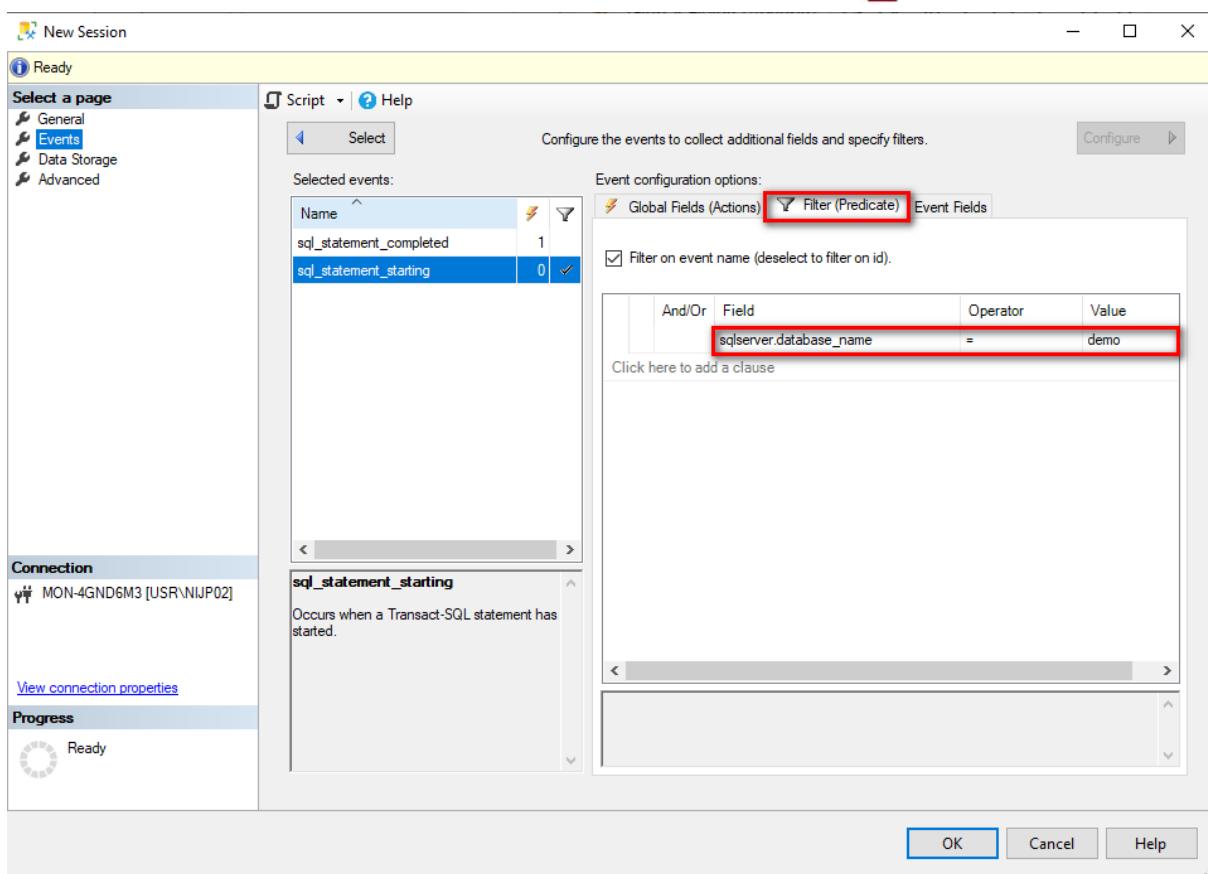
Extended Events is een opvolger van SQL Server Profiler die minder resources gebruikt. De sessie definities worden in de database opgeslagen zodat de databasebeheerder niet herhaaldelijk XE sessies hoeft te configureren maar on demand kan starten en stoppen.

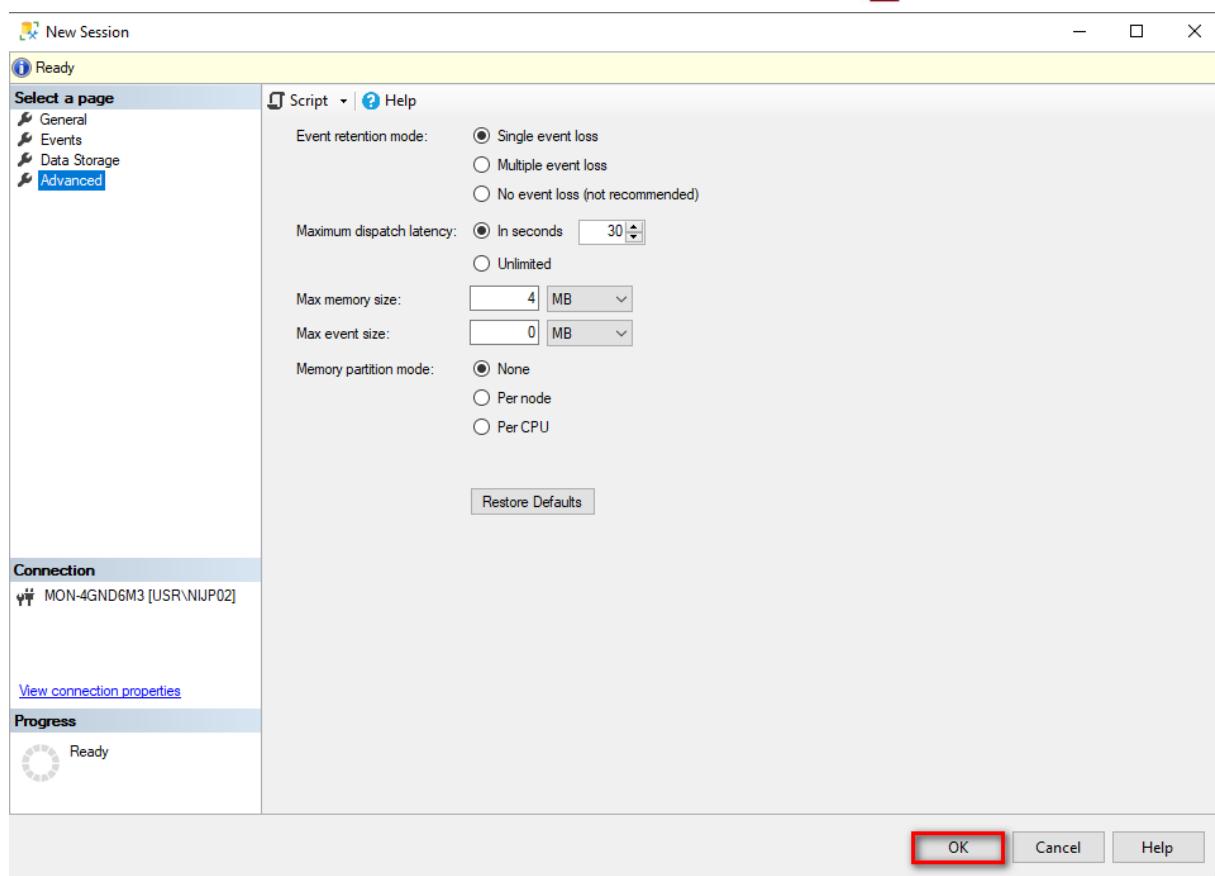
Extended Events kan opgestart worden vanuit de Object Explorer met Management -> Session -> New Session:

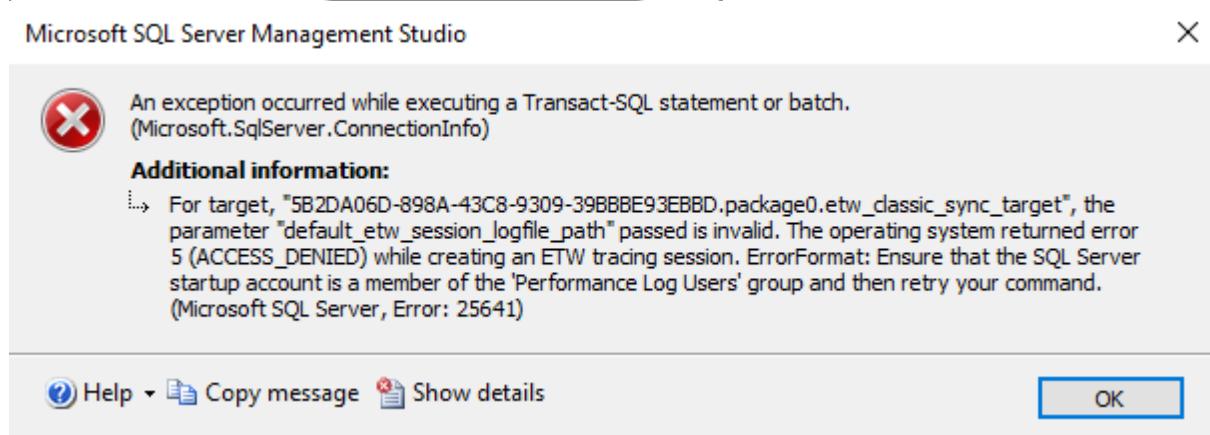
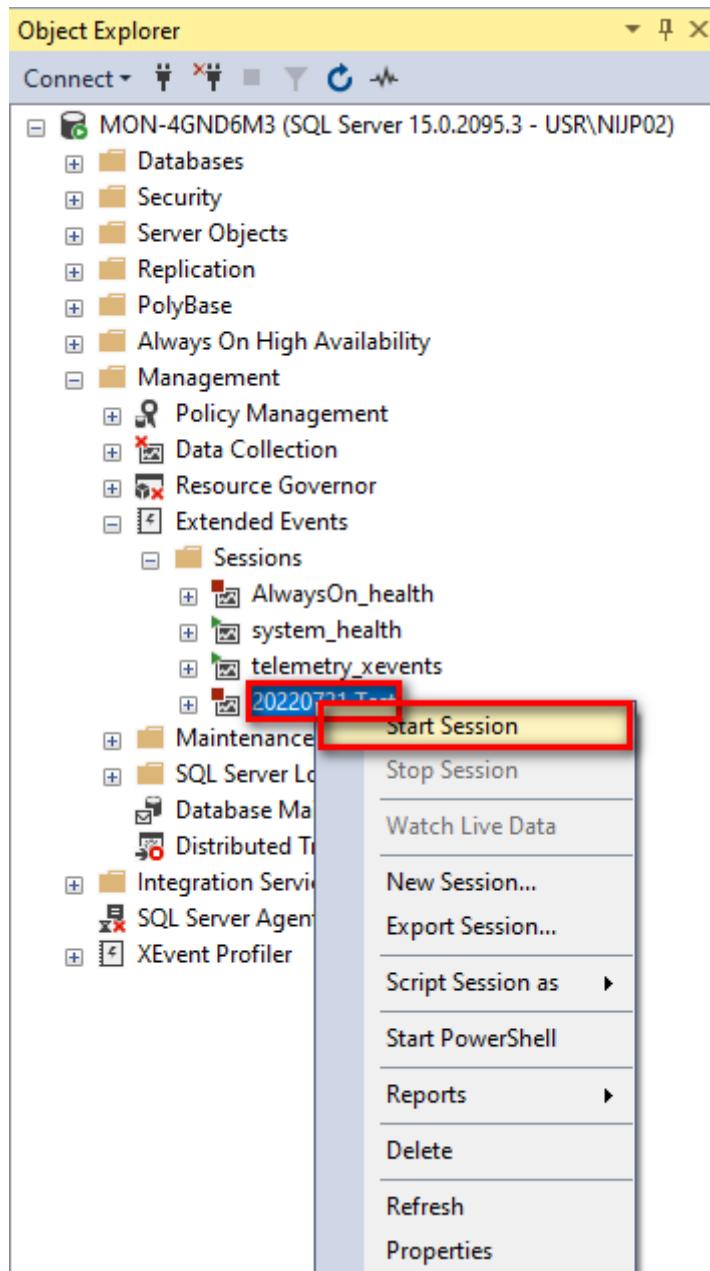


Onderstaande is onduidelijk en werkt niet









## Advanced Information

Message component:

- [-] All messages
  - [-] An exception occurred while exe
    - [...]
    - [...]
    - [...]
  - [-] For target, "5B2DA06D-898A-43C8-9309-39BBBE93EBBD.package0.etw\_classic\_sync\_target"
    - [...]
    - [...]
    - [...]
    - [...]
    - [...]
    - [...]

Details:

For target, "5B2DA06D-898A-43C8-9309-39BBBE93EBBD.package0.etw\_classic\_sync\_target", the parameter "default\_etw\_session\_logfile\_path" passed is invalid. The operating system returned error 5 (ACCESS\_DENIED) while creating an ETW tracing session. ErrorFormat: Ensure that the SQL Server startup account is a member of the 'Performance Log Users' group and then retry your command. (.Net SqlClient Data Provider)

For help, click: <https://docs.microsoft.com/sql/relational-databases/errors-events/mssqlserver-25641-database-engine-error>

Server Name: MON-4GND6M3  
 Error Number: 25641  
 Severity: 16  
 State: 0  
 Line Number: 1

Program Location:

at Microsoft.SqlServer.Management.Common.ConnectionManager.ExecuteNonQueryTSql(ExecuteTSqlAction action, Object execObject, DataSet fillDataSet,

[Close](#)

MON-4GND6M3 - system\_health: event\_file - Microsoft SQL Server Management Studio

File Edit View Project Tools Window Help

Object Explorer

MON-4GND6M3 - system\_health: event\_file

MON-4GND6M3 - sys...health: ring\_buffer

MON-4GND6M3 - sys...health: event\_file

Displaying 41923 Events

name	timestamp
sp_server_diagnostics_compo...	2022-07-19 22:51:57 9885618
sp_server_diagnostics_compo...	2022-07-19 22:51:57 9887455
sp_server_diagnostics_compo...	2022-07-19 22:51:57 9895533
sp_server_diagnostics_compo...	2022-07-19 22:51:57 9895671
scheduler_monitor_system_he...	2022-07-19 22:52:02 0260213
memory_broker_ring_buffer_re...	2022-07-19 22:52:29 6449762
memory_broker_ring_buffer_re...	2022-07-19 22:52:29 6449809
memory_broker_ring_buffer_re...	2022-07-19 22:52:29 6449840
memory_broker_ring_buffer_re...	2022-07-19 22:52:29 6449854
memory_broker_ring_buffer_re...	2022-07-19 22:52:29 6449863
memory_broker_ring_buffer_re...	2022-07-19 22:52:29 6449869
memory_broker_ring_buffer_re...	2022-07-19 22:52:29 6449881
memory_broker_ring_buffer_re...	2022-07-19 22:52:29 6449888
memory_broker_ring_buffer_re...	2022-07-19 22:52:29 6449895
memory_broker_ring_buffer_re...	2022-07-19 22:52:29 6449901
memory_broker_ring_buffer_re...	2022-07-19 22:52:29 6449906
memory_broker_ring_buffer_re...	2022-07-19 22:52:29 6449913
memory_broker_ring_buffer_re...	2022-07-19 22:52:29 7449644
memory_broker_ring_buffer_re...	2022-07-19 22:52:29 7449649

Event:

Details

Field	Value

niet

#### 25.4.11 Dynamic management

niet

#### 25.4.12 Data Collection

niet

#### 25.4.13 Query Store (TODO)

#### 25.4.14 Indexen en onderhoud

Wanneer gegevens niet gesorteerd zijn dan moet de database engine alle records (rijen) in de tabel scannen om te kijken of ze kandidaat zijn voor het resultaat van de query.

Een **index** is een object die sortering aanbrengt zoek mogelijkheden op niet gesorteerde data mogelijk maakt.

Een **heap** (ophoping) is een ongesorteerde tabel die geen clustered index heeft. Daar een primary key en unique key constraints intern een B-tree index maakt komt een pure heap weinig voor in een relationele data tabel omdat een tabel zonder constraint niet echt een relationele tabel is. Wanneer SQL Server een taak uitvoert op een heap dan scant hij alle data pages bij het ophalen van de gewenste rijen. Wanneer een record in een heap wordt geupdate, en de nieuwe rij past in de ruimte van de originele data pages dan gebeurt er niets maar wanneer de rij na de update niet meer past in de originele data pages dan wordt de rij uit de voorgaande pagina verwijderd en verplaatst naar een data page met voldoende ruimte en de oorspronkelijke record versie wordt gemarkeerd als **ghost record**. Wanneer een record wordt verwijderd dan wordt het record ook gemarkeerd als een ghost record. Fragmentatie komt op zich zelden voor en hangt af van meerdere ghost records.

Een **B-tree** index is een **Balanced Tree** Index. Een balanced tree is een boom die bestaat uit een **root** (wortel) en **intermediate nodes**, tussenliggende knooppunten (indien nodig) en **leaf nodes** die gesorteerde kopieën van index kolommen bevatten. Een record (rij) die opgeslagen is in een index wordt **index key** genoemd. De pages op eenzelfde niveau zijn onderling met elkaar verbonden. Elke page kent zijn voorganger en zijn opvolger, waardoor SQL Server snel en efficiënt naar de volgende pagina kan springen volgens de sortering in de index. Wanneer er een non-clustered B-tree index wordt gemaakt op de top van een heap, dan worden er leaf record gecreeerd las kopien van de waarden van de index kolom. Twee condities waarbij de index bruikbaar is, zijn wanneer de eerste kolom in de index gebruikt wordt in de WHERE clause en wanneer de WHERE clause selectief is (de verhouding tussen records die voldoen aan het filter t.o.v. het aantal record in de tabel klein is). Wanneer een query alleen kolommen bevat die aanwezig zijn in de non-clustered index dan hoeft SQL Server niet naar de tabel pages erbij te zoeken. Zo nodig volgt SQL Server de pointers in de leaf pages van de index naar de data pages. We kunnen ook **included columns** toevoegen aan de index. Deze included columns zijn niet geïndexeerd (dus de index key blijft klein), maar er hoeft dan niet naar de data pages te worden gekeken. Wanneer er een record wordt toegevoegd, dan kan dat

betekenen dat een page gesplitst moet worden in twee nieuwe pages. Bij een update waardoor de rij niet meer in de ruimte past wordt de rij verplaatst naar een andere page en wordt in de oorspronkelijke index page een pointer achtergelaten naar de nieuwe locatie omdat het goedkoper is dan de index key updaten. Wanneer een record wordt verwijderd dan wordt de record gemarkeerd als ghost record. Wanneer er veel ghost records zijn dan is de tabel gefragmenteerd en moet de tabel of index gedefragmenteerd worden.

Een clustered B-tree index

## Hoofdstuk 26 TRANSACTIONS EN CONCURRENCY (NEW)

Bij kleine bedrijven of in privé omgevingen worden veelal Stand-alone databases gebruikt waarbij er slechts één gebruiker tegelijk met de database kan werken (Single User database). Bij middelgrote en grote bedrijven zullen er op elk moment van de dag meerdere gebruikers tegelijk met de database werken (Multi User database), waardoor onderwerpen als transaction en locking belangrijke aspecten zijn.

### 26.1 Transactions

Een **transactie** is een verzameling SQL instructies die door één gebruiker ingevoerd wordt en waarbij aan het einde aangegeven wordt of alle mutaties permanent moeten worden of dat zij allen ongedaan gemaakt moeten worden. De grenzen van een transactie kunnen we expliciet of impliciet definiëren.

We kunnen in SQL Server een nieuwe transactie expliciet beginnen met de instructie:

**BEGIN TRANSACTION;**

We kunnen een transactie expliciet eindigen met de instructies:

**COMMIT [TRAN|TRANSACTION];** om aan te geven dat alle mutaties binnen de actuele transactie permanent moeten worden gemaakt. Het woord TRANSACTION mag ook weg gelaten worden.

**ROLLBACK [TRAN|TRANSACTION];** om aan te geven dat alle mutaties binnen de actuele transactie ongedaan moeten worden gemaakt.

Als we de grenzen van een transactie niet explicet definiëren dan behandelt SQL Server default (standaard) elk individuele statement als een transactie. Dat betekent dat SQL Server achter elk individuele statement een COMMIT uitvoert.

In SQL Server bestaat een **setting** (instelling) **IMPLICIT\_TRANSACTIONS** waarmee het default gedrag van de sessie voor commits aangepast kan worden:

#### **IMPLICIT\_TRANSACTIONS = ON**

Hierbij begint één van de volgende SQL instructies een nieuwe transactie (onzichtbare BEGIN TRANSACTION die eerst wordt uitgevoerd):

ALTER TABLE, FETCH, REVOKE, BEGIN TRANSACTION, GRANT, SELECT (\*zie onderstaande uitzonderingen), CREATE, INSERT, TRUNCATE TABLE, DELETE, MERGE, UPDATE, DROP, OPEN.

\*SELECT statements die niet uit een tabel selecteren starten géén impliciete transactie. B.v. SELECT GETDATE(), of SELECT 1, 'ABC' hebben géén transactie nodig.

Wanneer IMPLICIT\_TRANSACTIONS = ON dan is het dus niet nodig om een BEGIN TRANSACTION te definiëren maar is het wél nodig om het einde van de transactie met een COMMIT TRANSACTION of ROLLBACK TRANSACTION te definiëren.

#### **IMPLICIT\_TRANSACTIONS = OFF**

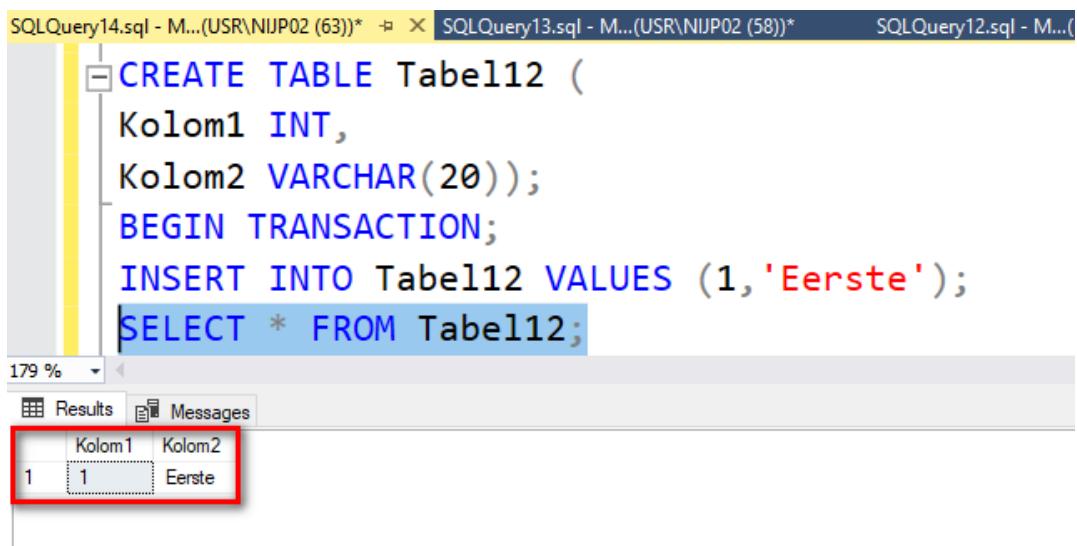
Dit is de default (standaard) instelling en wordt ook wel **Autocommit** genoemd.

Elke van de voorgaande SQL instructies (van IMPLICIT\_TRANSACTIONS = ON) worden begrensd door een onzichtbare BEGIN TRANSACTION én een onzichtbare COMMIT TRANSACTION instructie. SQL Server voert dus achter elke SQL statement automatisch een COMMIT uit.

### Voorbeeld 1:

We maken een tabel Tabel12 aan, beginnen een nieuwe transactie, doen een INSERT van één rij en doen een SELECT van alle gegevens in Tabel12. We zien dan dat de rij aan de tabel is toegevoegd.

```
USE Oefeningen;
CREATE TABLE Tabel12 (
Kolom1 INT,
Kolom2 VARCHAR(20));
BEGIN TRANSACTION;
INSERT INTO Tabel12 VALUES (1, 'Eerste');
SELECT * FROM Tabel12;
```



The screenshot shows the SQL Server Management Studio interface. A query window displays the following SQL code:

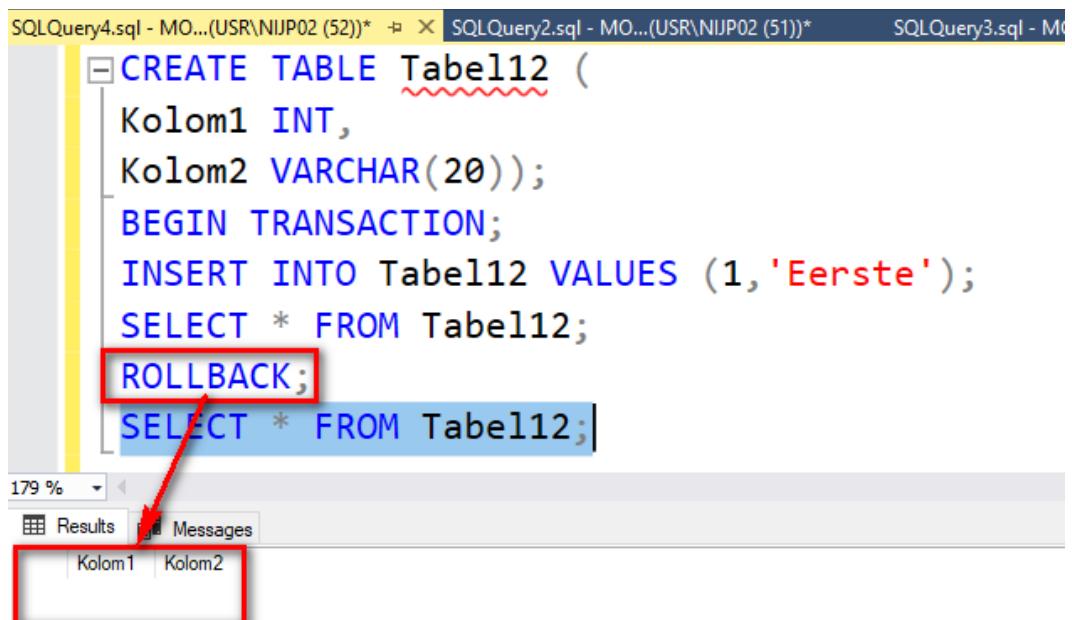
```
CREATE TABLE Tabel12 (
Kolom1 INT,
Kolom2 VARCHAR(20));
BEGIN TRANSACTION;
INSERT INTO Tabel12 VALUES (1, 'Eerste');
SELECT * FROM Tabel12;
```

The 'Results' tab is selected, showing the output:

	Kolom1	Kolom2
1	1	Eerste

Vervolgens voeren we een ROLLBACK uit en doen een SELECT van alle gegevens in Tabel12. We zien dan dat de toegevoegde rij aan de tabel weer is verwijderd.

```
ROLLBACK;
SELECT * FROM Tabel12;
```



The screenshot shows the SQL Server Management Studio interface. A query window displays the following SQL code, with a red box highlighting the ROLLBACK statement:

```
CREATE TABLE Tabel12 (
Kolom1 INT,
Kolom2 VARCHAR(20));
BEGIN TRANSACTION;
INSERT INTO Tabel12 VALUES (1, 'Eerste');
SELECT * FROM Tabel12;
ROLLBACK;
SELECT * FROM Tabel12;
```

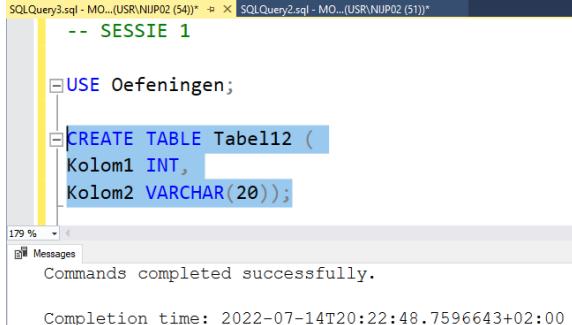
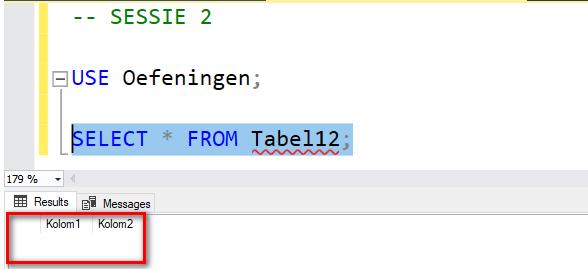
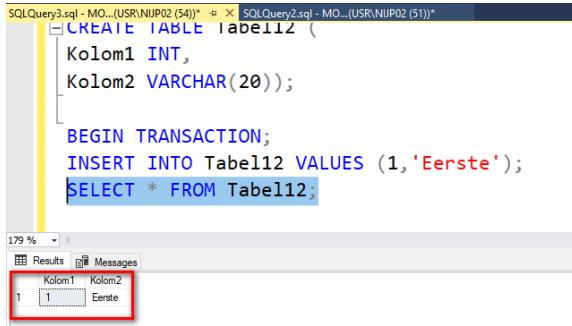
The 'Results' tab is selected, showing the output:

	Kolom1	Kolom2

Conclusie: De ROLLBACK maakt dus de acties binnen de actuele transactie ongedaan en brengt de situatie terug naar de situatie dat de BEGIN TRANSACTION was uitgevoerd.

### Voorbeeld 2:

Als we nu hetzelfde doen door twee SQL sessies op te starten (2x in SSMS op New Query klikken) en kijken wat de sessies aan de resultaten van elkaars activiteiten kunnen zien.

<pre>-- SESSIE 1</pre> <pre>USE Oefeningen; CREATE TABLE Tabel12 ( Kolom1 INT, Kolom2 VARCHAR(20));</pre>  <p>Commands completed successfully. Completion time: 2022-07-14T20:22:48.7596643+02:00</p>	<pre>-- SESSIE 2</pre> <pre>USE Oefeningen; SELECT * FROM Tabel12;</pre>  <p>De door Sessie 1 aangemaakte Tabel12 is dus zichtbaar in Sessie 2. Er zit nog géén gegevens in de Tabel12.</p> <pre>BEGIN TRANSACTION; INSERT INTO Tabel12 VALUES (1,'Eerste'); SELECT * FROM Tabel12;</pre> 
---	---

Na het starten van een nieuwe transactie wordt één rij ingevoerd. De ingevoerde rij is zichtbaar voor Sessie 1 m.b.v. de SELECT instructie.

```

COMMIT;
SELECT * FROM Tabel12;
SQLQuery2.sql - MO...(USR\NIJP02 (51))  SQLQuery3.sql - MO...(USR\NIJP02 (54))*
Kolom2 VARCHAR(20));

BEGIN TRANSACTION;
INSERT INTO Tabel12 VALUES (1,'Eerste');
SELECT * FROM Tabel12;
COMMIT;
SELECT * FROM Tabel12;

```

179 % Results Messages

Kolom1	Kolom2
1	Eerste

De COMMIT maakt de wijzigingen in de actuele transactie permanent en sluit de transactie af.

```

SELECT * FROM Tabel12;
SQLQuery2.sql - MON...(51)) Executing...*  SQLQuery3.sql - MO...(USR\NIJP02 (54))*
USE Oefeningen;

SELECT * FROM Tabel12;

SELECT * FROM Tabel12;

```

179 % Results Messages

Executing query... MON-4GND6M3 (15.0 RTM) | USR\NIJP02 (51) | Oefeningen | 0:

De door sessie 1 ingevoerde rij is echter niet zichtbaar voor Sessie 2 en de selectie van alle gegevens uit Tabel12 blijft executeren zonder einde (Executing Query...). De sessie staat te wachten en blijft hangen.

```

SQLQuery2.sql - MO...(USR\NIJP02 (51))  SQLQuery3.sql - MO...(USR\NIJP02 (54))*
-- SESSIE 2

USE Oefeningen;

SELECT * FROM Tabel12;

SELECT * FROM Tabel12;

```

179 % Results Messages

Kolom1	Kolom2
1	Eerste

Na de COMMIT in Sessie 1 wordt de door Sessie 1 ingevoerde rij in Tabel12 zichtbaar voor Sessie 2. De Executing Query status van de SELECT in Sessie 2 bleef dus hangen totdat de COMMIT in Sessie 1 werd gegeven.

Conclusie: De acties binnen de transactie in Sessie 1 blijft voor Sessie 2 niet zichtbaar totdat de wijzigingen permanent zijn gemaakt m.b.v. de COMMIT instructie. Alle wijzigingen binnen een transactie zijn binnen de transactie zelf zichtbaar maar blijven onzichtbaar voor alle andere gebruikers totdat de wijzigingen permanent zijn gemaakt m.b.v. de COMMIT. Wanneer er een ROLLBACK wordt uitgevoerd, dan blijven de wijzigingen binnen de transactie onzichtbaar voor de andere gebruikers.

M.b.v. **SAVEPOINT** kunnen we binnen een transactie punten markeren waar we m.b.v. een ROLLBACK naartoe kunnen gaan. M.b.v. een ROLLBACK naar een bepaald SAVEPOINT kunnen we een deel van de transactie ongedaan maken.

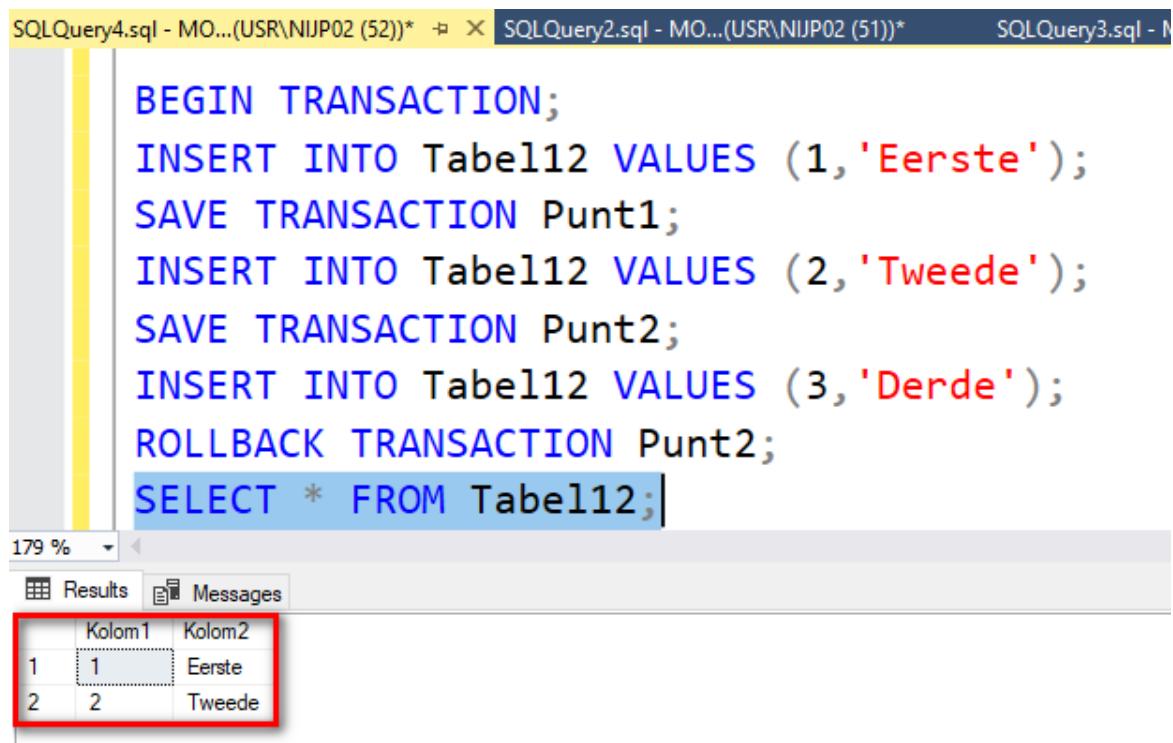
We kunnen een Savepoint aanmaken met de syntax:

**SAVEPOINT <SavepointNaam>;**

Voorbeeld:

We maken een nieuwe transactie aan waarbinnen we 2 Savepoints Punt1 en Punt2 aanmaken en voeren vervolgens een rollback uit naar Savepoint Punt2:

```
BEGIN TRANSACTION;
INSERT INTO Tabel12 VALUES (1, 'Eerste');
SAVE TRANSACTION Punt1;
INSERT INTO Tabel12 VALUES (2, 'Tweede');
SAVE TRANSACTION Punt2;
INSERT INTO Tabel12 VALUES (3, 'Derde');
ROLLBACK TRANSACTION Punt2;
SELECT * FROM Tabel12;
```



The screenshot shows a SQL Server Management Studio interface with three tabs at the top: 'SQLQuery4.sql - MO...', 'SQLQuery2.sql - MO...', and 'SQLQuery3.sql - M'. The main area contains the following SQL script:

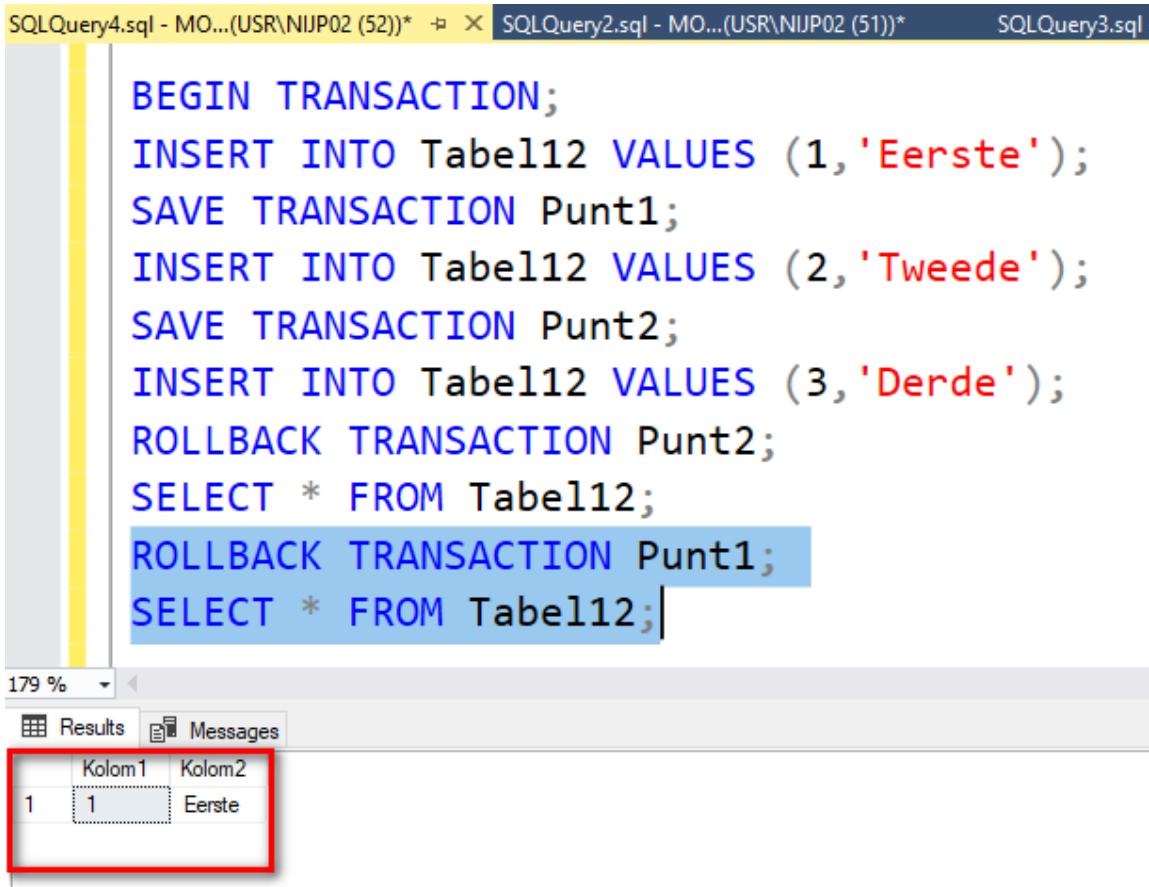
```
BEGIN TRANSACTION;
INSERT INTO Tabel12 VALUES (1, 'Eerste');
SAVE TRANSACTION Punt1;
INSERT INTO Tabel12 VALUES (2, 'Tweede');
SAVE TRANSACTION Punt2;
INSERT INTO Tabel12 VALUES (3, 'Derde');
ROLLBACK TRANSACTION Punt2;
SELECT * FROM Tabel12;
```

Below the script, the 'Results' tab is selected, displaying the following table:

	Kolom1	Kolom2
1	1	Eerste
2	2	Tweede

Daarna voeren we nog een rollback uit naar Savepoint Punt1:

```
ROLLBACK TRANSACTION Punt1;
SELECT * FROM Tabel12;
```



The screenshot shows a SQL Server Management Studio window with three tabs at the top: 'SQLQuery4.sql - MO...(USR\NIJP02 (52))\*' (highlighted in yellow), 'SQLQuery2.sql - MO...(USR\NIJP02 (51))\*', and 'SQLQuery3.sql'. The main pane contains the following SQL code:

```
BEGIN TRANSACTION;
INSERT INTO Tabel12 VALUES (1, 'Eerste');
SAVE TRANSACTION Punt1;
INSERT INTO Tabel12 VALUES (2, 'Tweede');
SAVE TRANSACTION Punt2;
INSERT INTO Tabel12 VALUES (3, 'Derde');
ROLLBACK TRANSACTION Punt2;
SELECT * FROM Tabel12;
ROLLBACK TRANSACTION Punt1;
SELECT * FROM Tabel12;
```

The results pane shows a table with two columns: 'Kolom1' and 'Kolom2'. The data is:

	Kolom1	Kolom2
1	1	Eerste

Conclusie: Na de rollback naar Savepoint Punt2 is de derde insert teruggedraaid en zitten de eerste en tweede inserts nog wel in de tabel. Na de rollback naar Savepoint Punt1 is ook de tweede insert teruggedraaid en zit alleen de eerste insert nog in de tabel.

We kunnen zo nodig het aantal transacties dat opgestart is opvragen met:

```
PRINT @@TRANCOUNT
```

Voorbeeld 1:

SQLQuery30.sql - M...r (USR\NIJP02 (57))\* × SQLQuery29.sql - M...(USR\NIJP02 (52))\*

```

PRINT @@TRANCOUNT
BEGIN TRAN
    PRINT @@TRANCOUNT
    BEGIN TRAN
        PRINT @@TRANCOUNT
        COMMIT
        PRINT @@TRANCOUNT
    COMMIT
    PRINT @@TRANCOUNT

```

79 %

Messages

```

0
1
2
1
0

```

Completion time: 2022-07-19T10:33:39.1903566+02:00

Conclusie: Elke BEGIN TRANS verhoogt de waarde van @@TRANCOUNT met 1 en elke COMMIT vermindert de waarde van @@TRANCOUNT met 1.

Voorbeeld 2:

SQLQuery31.sql - M...r (USR\NIJP02 (55))\* × SQLQuery30.sql - M...r (USR\NIJP02 (57))\* SQLQuery29.sql - M...r (USR\NIJP02 (52))\*

```

PRINT @@TRANCOUNT
BEGIN TRAN
    PRINT @@TRANCOUNT
    BEGIN TRAN
        PRINT @@TRANCOUNT
    ROLLBACK
    PRINT @@TRANCOUNT

```

179 %

Messages

```

0
1
2
0

```

Completion time: 2022-07-19T10:35:27.9244541+02:00

Conclusie: Elke BEGIN TRANS verhoogt de waarde van @@TRANCOUNT met 1 en de ROLLBACK draait alle transacties terug en zet de waarde van @@TRANCOUNT weer op 0.

## 26.2 Locking

Locking is een mechanisme waarbij een tabel of rij wordt geblokkeerd voor andere gebruikers zodat de andere gebruikers niet meer in staat zijn om deze rij te benaderen. Alleen de gebruiker die de rij geblokkeerd heeft kan de rij benaderen. De blokkades verdwijnen weer wanneer de transactie beëindigd wordt. De levensduur van een blokkade is dus nooit langer dan die van de transactie waarin de blokkade is gecreëerd. Andere gebruikers zullen moeten wachten tot de eerste gebruiker, die de lock heeft geplaatst, met de rij klaar is.

Er bestaan verschillende soorten locks, waarvan de **Shared Lock (S)** en de **Exclusive Lock (X)** nu de meest belangrijkste zijn. De Shared Lock wordt ook wel **Read Lock** genoemd en de Exclusive Lock wordt ook wel **Write lock** genoemd.

Wanneer je gegevens wilt modificeren, dan zal jouw transactie een Exclusive Lock aanvragen op de data resource. Wanneer de Exclusieve Lock is toegekend dan blijft deze tot het einde van de transactie staan. Dus bij een single-statement transaction tot het einde van het statement of bij een multi-statement transaction tot een COMMIT TRANSACTION of ROLLBACK TRANSACTION commando.

Exclusive Locks worden exclusief genoemd omdat je één exclusive lock kan verwerven op een resource als een andere transactie een lock houdt op de resource, en er één lock kan worden verworven op een resource wanneer een andere transactie een exclusive lock houdt op de resource. Of een andere transactie dezelfde rijen kan lezen hangt af van zijn zogenaamde isolation level. Isolation levels vallen echter buiten het bestek van deze reader.

In SQL Server is de default **READ COMMITTED**. Wanneer je probeert om data te lezen, dan vraagt jouw transaction een SHARED LOCK op de data resource en laat deze lock los (release) zodra de leesactie is uitgevoerd. Deze lock wordt SHARED genoemd omdat meerdere transacties gelijktijdig een shared lock op dezelfde data resource kunnen hebben.

SQL Server kan verschillende resources locken zoals rows, pages, objects, database, etc. Rijen zitten binnen pages en pages zijn fysieke datablokken die tabel of index data bevatten. Om een **lock** op een bepaald type resource te plaatsen moet de transactie eerst een **intent lock** van dezelfde mode plaatsen op hogere niveaus. Bijvoorbeeld om een exclusive lock op een rij te kunnen krijgen moet de transactie eerst een intent exclusive lock krijgen op de tabel en op de page waarin de rij zich bevindt. De intent locks interfereren (samen of tegenwerken) niet met lock aanvragen op lager niveau. Bijvoorbeeld een intent lock op een page voorkomt niet dat andere transacties incompatible lock modes op rijen binnen de page verwerven.

De lock interactie tussen de transacties wordt **lock compatibility** genoemd en kan worden weergegeven in een tabel, waarbij de kolommen de granted lock modes weergeven en de rijen de requested lock modes (No = denied, Yes = accepted):

Requested mode	Granted Exclusive (X)	Granted Shared (S)	Granted Intent Exclusive (IX)	Granted Intent Shared (IS)
Exclusive	No	No	No	No
Shared	No	Yes	No	Yes

Intent Exclusive	No	No	Yes	Yes
Intent Shared	No	Yes	Yes	Yes

SQL Server bepaalt dynamisch welke resources type gebruikt wordt voor de lock. Enerzijds is het beste dat hij alleen de rijen die gewijzigd worden lockt. Anderzijds gebruiken locks memory resources en geven ze intern management overhead. Wanneer SQL Server inschat dat de transactie een klein aantal rijen zal wijzigen dan neigt hij naar row locks en bij een groter aantal rijen zal hij neigen naar page locks.

Met de optie **ALTER TABLE ... LOCK ESCALATION** is het mogelijk om het gedrag per tabel te veranderen op tabel niveau (default) of op partitie niveau (tabel kan opgedeeld zijn in kleineren partities).

Wanneer een transactie een lock houdt op een resource en een andere transactie een incompatible lock aanvraagt op dezelfde resource dan wordt die aanvraag geblokkeerd (**blocked**) en komt de aanvragen in een **WAIT** status.

Blocking is normaal in een systeem, zolang als de aanvragers tevreden zijn met de hoeveelheid tijd. Wanneer echter een request te lang duurt, dan is het voor de databasebeheerder mogelijk noodzakelijk om te troubleshooten in de blocking situatie. M.b.v. de dynamic management view **sys.dm\_tran\_locks** kan de lock en de wait informatie worden verkregen:

**SELECT**

```
request_session_id AS sid,
resource_type AS restype,
resource_database_id AS dbid,
DB_NAME(resource_database_id) AS dbname,
resource_description AS res,
resource_associated_entity_id AS resid,
request_mode AS mode,
request_status AS status
```

FROM **sys.dm\_tran\_locks**;

Voorbeeld:

Twee transacties gebruiken dezelfde rij met ProductID = 2 van tabel Products

Sessie 1:

```
USE NORTHWIND;
BEGIN TRANSACTION;
UPDATE Products
SET UnitPrice += 1.00
WHERE productid= 2;
```

(1 row affected)

Completion time: 2022-07-19T16:16:30.0439947+02:00

179 % 179 %

Query executed successfully. MON-4GND6M3 (15.0 RTM) USR\NJP02 (60) Northwind 00:00:00 0 rows

Sessie 2:

```
USE Northwind;
SELECT ProductID, UnitPrice
FROM Products
WHERE ProductID = 2;
```

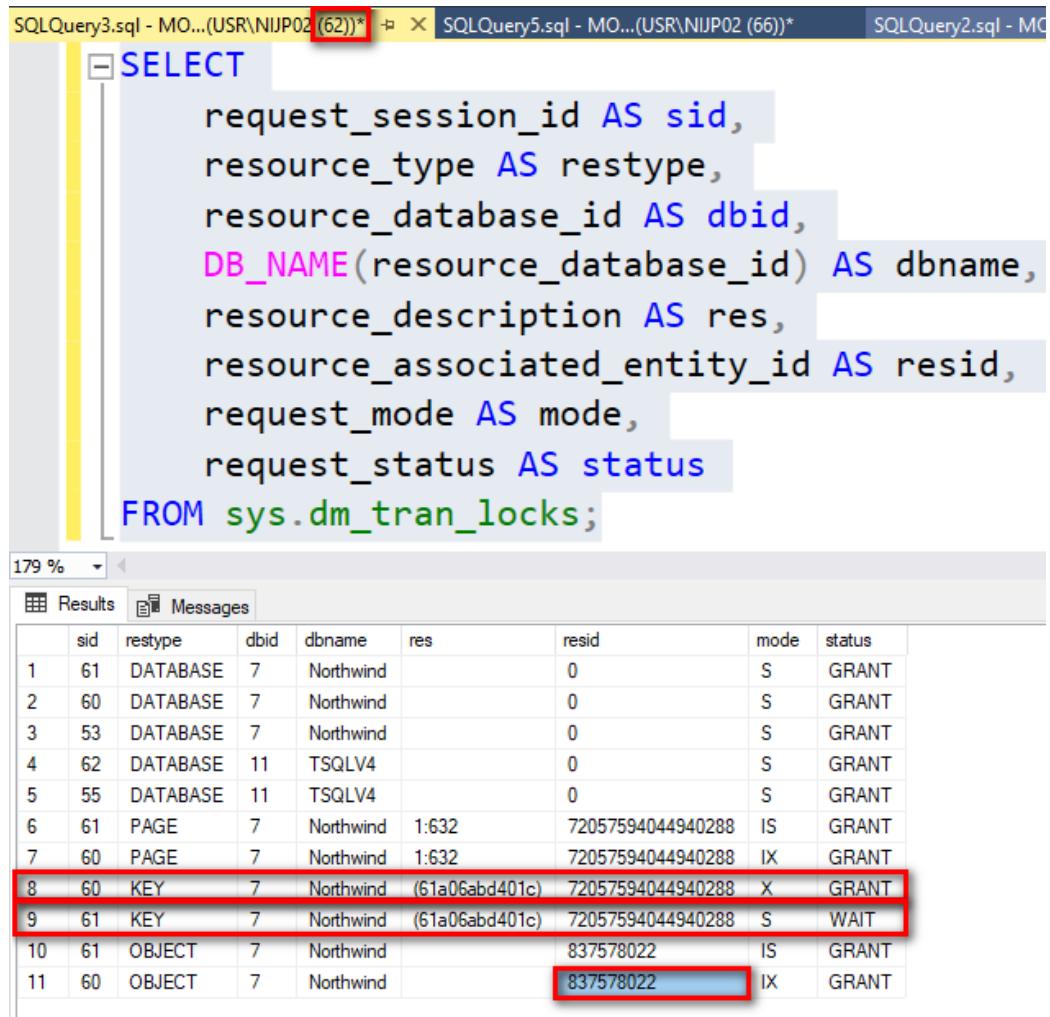
179 % 179 %

Results Messages

Executing query... MON-4GND6M3 (15.0 RTM) USR\NJP02 (61) Northwind 00:30:46 0 rows

De eerste sessie (sessionId 60) is succesvol. De tweede sessie (sessionId 61) is geblokkeerd en komt in een WAIT status. Met een derde sessie (sessionId 62) kunnen we de situatie van de locks en waits bekijken.

Sessie 3:



```

SELECT
    request_session_id AS sid,
    resource_type AS restype,
    resource_database_id AS dbid,
    DB_NAME(resource_database_id) AS dbname,
    resource_description AS res,
    resource_associated_entity_id AS resid,
    request_mode AS mode,
    request_status AS status
FROM sys.dm_tran_locks;

```

	sid	restype	dbid	dbname	res	resid	mode	status
1	61	DATABASE	7	Northwind		0	S	GRANT
2	60	DATABASE	7	Northwind		0	S	GRANT
3	53	DATABASE	7	Northwind		0	S	GRANT
4	62	DATABASE	11	TSQLV4		0	S	GRANT
5	55	DATABASE	11	TSQLV4		0	S	GRANT
6	61	PAGE	7	Northwind	1:632	72057594044940288	IS	GRANT
7	60	PAGE	7	Northwind	1:632	72057594044940288	IX	GRANT
8	60	KEY	7	Northwind	(61a06abd401c)	72057594044940288	X	GRANT
9	61	KEY	7	Northwind	(61a06abd401c)	72057594044940288	S	WAIT
10	61	OBJECT	7	Northwind		837578022	IS	GRANT
11	60	OBJECT	7	Northwind		837578022	IX	GRANT

In het overzicht is te zien dat de sessie met sessionId 61 in een wait status staat te wachten op de lock van de sessie met sessionId 60 op resourceId 72057594044940288. En tevens zien we dat SessionId 60 een Intent Exclusive Lock heeft op Object 837578022, waarvan we de naam van het object kunnen opvragen met de functie **OBJECT\_NAME**:

```
PRINT OBJECT_NAME(837578022);
```



```

PRINT OBJECT_NAME(837578022);

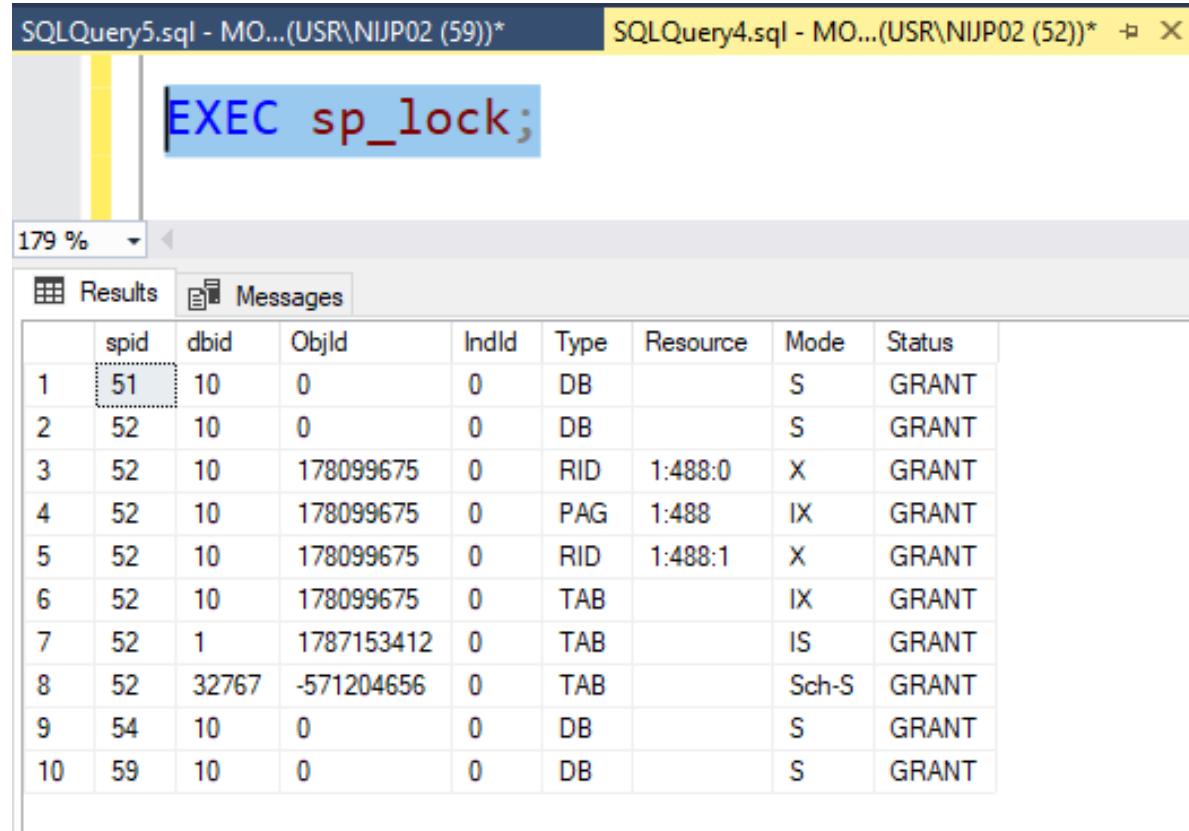
```

Products
----------

Completion time: 2022-07-19T17:19:52.1012902+02:00

NB: De actuele locks in een database kan ook worden gevonden met de System Procedure **sp\_lock**:

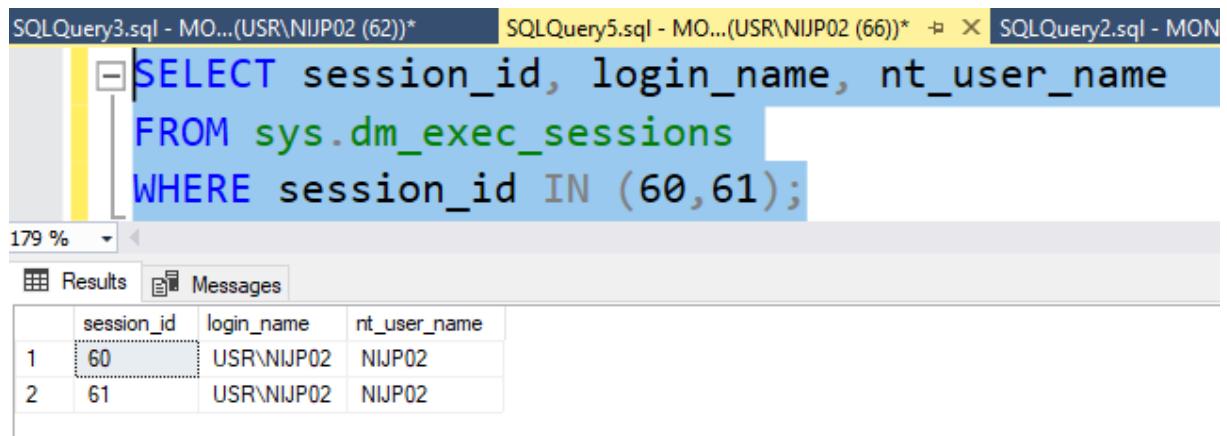
```
EXEC sp_lock;
```



	spid	dbid	ObjId	IndId	Type	Resource	Mode	Status
1	51	10	0	0	DB		S	GRANT
2	52	10	0	0	DB		S	GRANT
3	52	10	178099675	0	RID	1:488:0	X	GRANT
4	52	10	178099675	0	PAG	1:488	IX	GRANT
5	52	10	178099675	0	RID	1:488:1	X	GRANT
6	52	10	178099675	0	TAB		IX	GRANT
7	52	1	1787153412	0	TAB		IS	GRANT
8	52	32767	-571204656	0	TAB		Sch-S	GRANT
9	54	10	0	0	DB		S	GRANT
10	59	10	0	0	DB		S	GRANT

We weten nu welke sessie een lock houdt op de tabel Products en kunnen we de persoon benaderen om te vragen waarom zijn sessie de lock zo lang vast houdt. We kunnen dan de login naam van de persoon van de sessie opvragen met de System Catalog View **sys.dm\_exec\_sessions**:

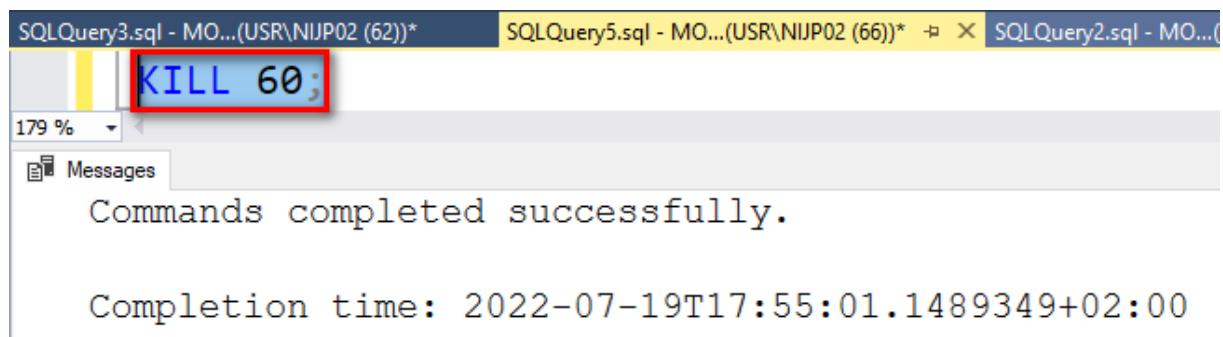
```
SELECT session_id, login_name, nt_user_name
FROM sys.dm_exec_sessions
WHERE session_id IN (60,61);
```



	session_id	login_name	nt_user_name
1	60	USR\NIJP02	NIJP02
2	61	USR\NIJP02	NIJP02

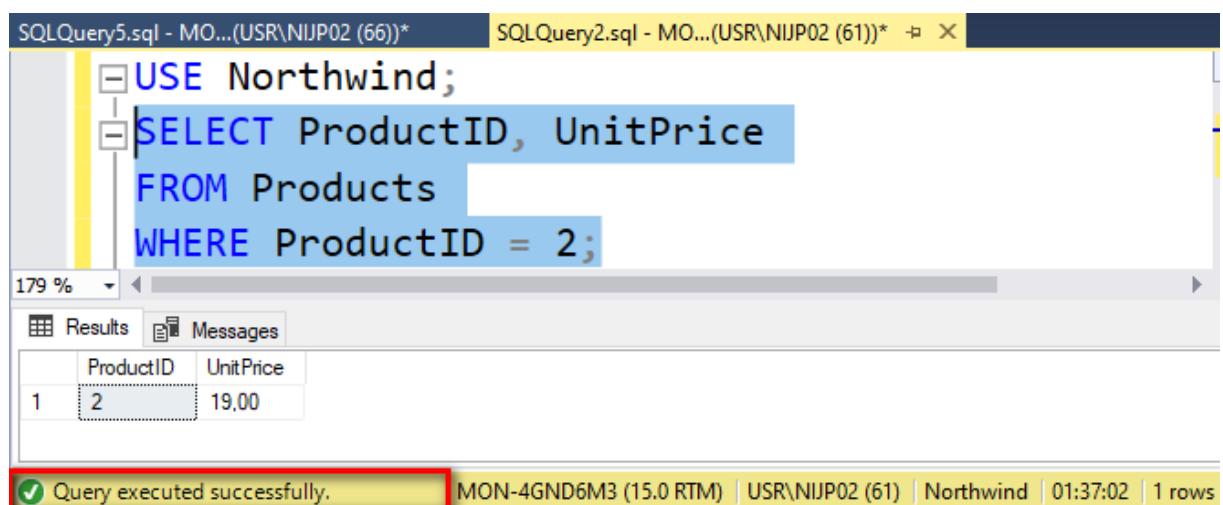
Vaak blijkt dat de persoon een actie was gestart maar even was weg gelopen en vergeten was om de sessie af te sluiten. Aan de gebruiker kan dan worden gevraagd om de sessie af te ronden of af te breken.

Soms is de betreffende persoon reeds naar huis terwijl zijn PC nog in de database ingelogd is en een lock vast houdt. In dat geval willen we de sessie afbreken, wat we kunnen doen met het commando **KILL**:



The screenshot shows the SQL Server Management Studio interface. The top tab bar has three tabs: 'SQLQuery3.sql - MO... (USR\NIJP02 (62))\*', 'SQLQuery5.sql - MO... (USR\NIJP02 (66))\*', and 'SQLQuery2.sql - MO... (USR\NIJP02 (61))\*'. The main query window contains the command 'KILL 60;'. Below the query window, the 'Messages' tab is selected, showing the output: 'Commands completed successfully.' and 'Completion time: 2022-07-19T17:55:01.1489349+02:00'.

Direct na het killen van de sessie met sessionID 60 kan de sessie met sessionID 61 zijn actie voortzetten:



The screenshot shows the SQL Server Management Studio interface. The top tab bar has two tabs: 'SQLQuery5.sql - MO... (USR\NIJP02 (66))\*' and 'SQLQuery2.sql - MO... (USR\NIJP02 (61))\*'. The main query window contains the following SQL code:  
`USE Northwind;  
SELECT ProductID, UnitPrice  
FROM Products  
WHERE ProductID = 2;`

The results pane shows a single row of data:  

ProductID	UnitPrice
1	2

The status bar at the bottom of the screen displays the message 'Query executed successfully.' with a green checkmark icon.

M.b.v. de System Catalog Views en de System Stored Functions kunnen we nog veel dieper spitten wanneer de situatie complexer is dan in dit voorbeeld, echter valt dat buiten het bestek van deze reader. Voor nu voldoet het dat we weten hoe we een lock en een wait kunnen vinden, de login naam van de sessie kunnen terugvinden en de sessie uit de database kunnen killen.

Default gebruikt SQL Server voor leesacties SHARED locks en voor schrijfacties EXCLUSIVE locks. Het is binnen SQL Server niet mogelijk om het gedrag van schrijfacties aan te passen, maar het is wel mogelijk om het gedrag bij leesacties aan te passen m.b.v **Isolation Levels**. Dit valt echter buiten het bestek van deze reader. Voor nu voldoet het dat we weten dat leesacties shared locks vragen en dat schrijfacties exclusive locks vragen.

Wanneer twee gebruikers op elkaar staan te wachten dan wordt dat een **Deadlock** genoemd. Bij voorbeeld wanneer gebruiker 1 eerst rij1 wil gebruiken en daarna rij2 terwijl een andere gebruiker gelijktijdig eerst rij2 en daarna rij1 wil gebruiken, dan kunnen ze in een situatie terecht komen dat beiden reeds een rij hebben gebruikt maar voor de tweede rij op elkaar moeten wachten. In deze situatie komt dan geen einde totdat één van de gebruikers zijn actie afbreekt en terugdraait. Wanneer SQL Server een deadlock ontdekt dan zal hij één van de transacties kiezen en dwingen om alle acties terug te draaien. Mits niet anders gespecificeerd zal SQL Server de transactie die het minste hoeveelheid werk heeft verricht (op basis van de transaction log) afbreken omdat de rollback van die transactie de goedkoopste optie is.

Het is mogelijk om m.b.v. **DEADLOCK\_PRIORITY** prioriteiten te gebruiken om het gedrag van SQL Server te manipuleren, echter valt dat buiten het bestek van deze reader. Voor nu voldoet het dat we weten dat SQL Server zelf een transactie zal terugdraaien wanneer er een deadlock situatie ontstaat.

## Hoofdstuk 27 SYSTEM CATALOG VIEWS en SYSTEM STORED PROCEDURES (NEW)

### 27.1 System Catalog

De **System Catalog** (systeemcatalogus) is een groep tabellen en views die de essentiële details van de database bevatten. Deze gegevens worden de **Metadata** (Metagegevens) van de database genoemd en beschrijven de context, de inhoud en de structuur van de informatieobjecten.

Elke database bevat een system catalog en de informatie in de system catalog specificert het framework (raamwerk) van de database. Alle gegevens die we zien in de Graphical User Interface van SSMS komen uit de System Catalog. Zoals de gegevens over de databases, tabellen, kolommen, Constraints, Indexen, Users, Rollen, Schema's, etc. De Database Engine/RDBMS gebruikt regelmatig de systeemcatalogus voor informatie die essentieel is om het systeem goed te laten functioneren.

De systeemtabellen van de hoofddatabase (**master**) vormen de **System Catalog** en de systeemtabellen van een door de gebruiker gedefinieerde database vormen de **Database Catalog**. De systeembasistabellen komen slechts één keer voor in het gehele systeem en de database basistabellen komen één keer voor in elke database, inclusief in de hoofddatabase.

In alle relationele databasesystemen hebben systeembasistabellen dezelfde logische structuur als basistabellen. Als gevolg hiervan kunnen dezelfde SQL-instructies die worden gebruikt om informatie in de basistabellen op te halen, ook worden gebruikt om informatie in systeembasistabellen op te halen. De systeembasistabellen zijn echter niet rechtstreeks toegankelijk, maar kunnen worden benaderd via **Catalog Views** en **System Stored Procedures**.

### 27.2 Catalog Views

Er bestaan zeer veel Catalog Views waar we gegevens uit kunnen halen. Hieronder volgen een aantal voorbeelden.

Gegevens over alle objecten in de courante database plus objecten in de master database:

```
SELECT * FROM sys.all_objects;
```

Gegevens over alle objecten in de courante database:

```
SELECT * FROM sys.objects;
```

Gegevens over alle tabellen in de courante database:

```
SELECT * FROM sys.tables;
```

Gegevens over alle kolommen in de courante database:

```
SELECT * FROM sys.columns;
```

Gegevens over alle kolommen van een tabel Orders in de courante database Northwind:

```
USE Northwind;
SELECT tables.object_id, tables.name AS 'Table Name',
columns.column_id, columns.name AS 'Column Name'
FROM sys.tables, sys.columns
WHERE tables.object_id = columns.object_id
```

AND Tables.Name = 'Orders';

MON-4GND6M3 - Activity Monitor    SQLQuery10.sql - M...(USR\NIJP02 (52))\*    SQLQuery9.sql - MO...(USR\NIJP02 (54))\*

```
SELECT tables.object_id, tables.name AS 'Table Name',
       columns.column_id, columns.name AS 'Column Name'
  FROM sys.tables, sys.columns
 WHERE tables.object_id = columns.object_id
   AND Tables.Name = 'Orders';
```

179 %

	object_id	Table Name	column_id	Column Name
1	805577908	Orders	1	OrderID
2	805577908	Orders	2	CustomerID
3	805577908	Orders	3	EmployeeID
4	805577908	Orders	4	OrderDate
5	805577908	Orders	5	RequiredDate
6	805577908	Orders	6	ShippedDate
7	805577908	Orders	7	ShipVia
8	805577908	Orders	8	Freight
9	805577908	Orders	9	ShipName
10	805577908	Orders	10	ShipAddress
11	805577908	Orders	11	ShipCity
12	805577908	Orders	12	ShipRegion
13	805577908	Orders	13	ShipPostalCode
14	805577908	Orders	14	ShipCountry

Op deze manier kunnen we m.b.v. JOINS over de juiste catalog views gegevens opvragen over alle objecten in de database.

Een overzicht van de aanwezige System Catalog Views is te verkrijgen m.b.v. de query:

```
SELECT * FROM sys.all_objects WHERE schema_id = 4 and type = 'V';
```

### 27.3 System Stored Procedures

Er bestaan ook System Stored Procedures om een set gegevens over meerdere objecten samen te voegen in voor gedefinieerde rapporten. Hieronder volgen een aantal voorbeelden.

Om gegevens over de tabel Orders in de database Northwind te verkrijgen:

```
USE Northwind;
EXEC sp_tables Orders;
```

MON-4GND6M3 - Activity Monitor    SQLQuery10.sql - M...(USR\NIJP02 (52))\*    SQLQuery9.sql - MO...(USR\NIJP02 (54))\*

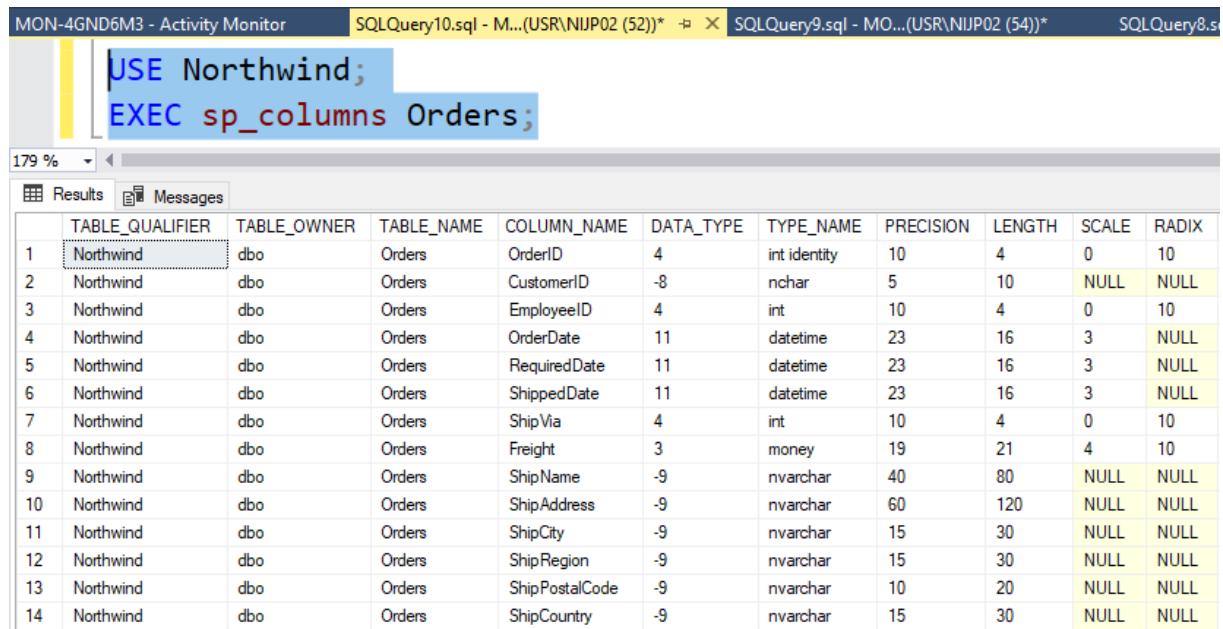
```
USE Northwind;
EXEC sp_tables Orders;
```

179 %

	TABLE_QUALIFIER	TABLE_OWNER	TABLE_NAME	TABLE_TYPE	REMARKS
1	Northwind	dbo	Orders	TABLE	NULL

Om gegevens van de kolommen van een tabel Orders in de database Northwind te verkrijgen:

```
USE Northwind;
EXEC sp_columns Orders;
```



	TABLE_QUALIFIER	TABLE_OWNER	TABLE_NAME	COLUMN_NAME	DATA_TYPE	TYPE_NAME	PRECISION	LENGTH	SCALE	RADIX
1	Northwind	dbo	Orders	OrderID	4	int identity	10	4	0	10
2	Northwind	dbo	Orders	CustomerID	-8	nchar	5	10	NULL	NULL
3	Northwind	dbo	Orders	EmployeeID	4	int	10	4	0	10
4	Northwind	dbo	Orders	OrderDate	11	datetime	23	16	3	NULL
5	Northwind	dbo	Orders	RequiredDate	11	datetime	23	16	3	NULL
6	Northwind	dbo	Orders	ShippedDate	11	datetime	23	16	3	NULL
7	Northwind	dbo	Orders	ShipVia	4	int	10	4	0	10
8	Northwind	dbo	Orders	Freight	3	money	19	21	4	10
9	Northwind	dbo	Orders	ShipName	-9	nvarchar	40	80	NULL	NULL
10	Northwind	dbo	Orders	ShipAddress	-9	nvarchar	60	120	NULL	NULL
11	Northwind	dbo	Orders	ShipCity	-9	nvarchar	15	30	NULL	NULL
12	Northwind	dbo	Orders	ShipRegion	-9	nvarchar	15	30	NULL	NULL
13	Northwind	dbo	Orders	ShipPostalCode	-9	nvarchar	10	20	NULL	NULL
14	Northwind	dbo	Orders	ShipCountry	-9	nvarchar	15	30	NULL	NULL

## 27.4 Gebruik van Catalog Views en System Stored Procedures

Voor dagelijks gebruik is het handig en snel om via de boom in de Graphical User Interface te zoeken naar informatie over de objecten in de database.

Soms wil de databasebeheerder periodiek SQL scripts op de databases uitvoeren om bijvoorbeeld dagelijks een overzicht te verkrijgen van de gegevens van de objecten in de databases. Iedere databasebeheerder wil andere gegevens zien en iedere databasebeheerder heeft een eigen voorkeur van hoe een rapport er uit moet zien. De databasebeheerder zal daarom zelf SQL scripts moeten maken die de gewenste gegevens uit de Catalog Views haalt.

De databasebeheerder moet daarom wegwijs zijn in de Catalog Views en System Stored Procedures. Hij hoeft niet alle views en procedures uit het hoofd te kennen, maar moet wel weten waar hij de gegevens daarover kan vinden en hoe hij ze kan gebruiken.

Een overzicht van de aanwezige System Stored Procedures is te verkrijgen m.b.v. de query:

```
SELECT * FROM sys.all_objects WHERE schema_id = 4 and type = 'P';
```

Een goed begin om naar de Catalog Views en Procedures te zoeken is:

<https://docs.microsoft.com/en-us/sql/relational-databases/system-catalog-views/catalog-views-transact-sql?view=sql-server-ver16>

<https://docs.microsoft.com/en-us/sql/relational-databases/system-catalog-views/querying-the-sql-server-system-catalog-faq?view=sql-server-ver16>

Maar er zijn meerdere sites waar goede informatie beschikbaar is, welke weer te vinden zijn via Google...

## Hoofdstuk 28 SQL SERVER OP LINUX

Bij hele kleine bedrijven kan een database systeem geïnstalleerd zijn op een PC en wordt de database door slechts één gebruiker tegelijk gebruikt. Bij kleine en middelgrote bedrijven worden databases veelal op een server geïnstalleerd en maken meerdere gebruikers tegelijkertijd gebruik van de database. Vaak is deze server dan een combinatie van een file server (om bestanden op te slaan), een applicatie server (om applicaties op te draaien) en een database server (om een database management system op te draaien). Bij grote bedrijven worden databases op een mainframe of op aparte specifieke database servers geïnstalleerd. File servers, applicatie servers en database servers zijn dan van elkaar gescheiden zodat de activiteiten op de servers elkaar niet kunnen beïnvloeden. Wanneer we bijvoorbeeld een file server, een applicatie server en een databaseserver zouden combineren op één gezamenlijke fysieke server dan zou een back-up proces van de bestanden in de fileserver de performance van de applicatieserver en de databaseserver kunnen beïnvloeden of zou een zware activiteit op de applicatieserver de performance op de databaseserver kunnen beïnvloeden.

Databaseservers worden sinds de jaren zestig geïnstalleerd op Mainframes of op Midrange systemen met Unix of OpenVMS als operating system. Tegenwoordig worden steeds meer kleine en middelgrote databasesystemen vanwege de kostprijs geïnstalleerd op Linux. De kans dat jij als beheerder in de praktijk te maken gaat krijgen met een database op Linux is daarom redelijk groot. Om die reden zullen wij in dit hoofdstuk een kijkje nemen naar SQL-Server op Linux en zullen met SSMS een verbinding maken met SQL Server op de Linux Virtual Machine.

UNIX is een besturingssysteem die reeds ontwikkeld was sinds 1969. Veel server leveranciers leveren nog hun eigen UNIX variant op hun eigen processoren zoals AIX van IBM, HP-UX van HP en er zijn nog veel andere UNIX varianten zoals OpenServer van SCO voor Intel processoren. Linux is een Unix variant op een Intel processor die sinds 1984 is ontwikkeld die moet gaan lijken op UNIX maar minder duur moet zijn. Door de kostprijs van Linux, de kostprijs van de Intel processoren en OpenSource is Linux in de laatste decennia steeds meer populair geworden en is het inmiddels een betrouwbaar operating system met goede ondersteuning van organisaties als Redhat en Suse.

Linux is een UNIX variant en lijkt dan ook voor de gebruikers erg veel op de andere UNIX systemen. Voor de gebruikers zijn de UNIX en de Linux commando's vrijwel hetzelfde. Hoe dieper in de techniek, des te meer de verschillende UNIX varianten van elkaar verschillen, onder andere omdat ze op verschillende processoren draaien.

Jij zult je misschien afvragen waarom er nog steeds gebruik wordt gemaakt van de techniek uit de jaren zestig van de vorige eeuw, meer dan een halve eeuw oud met onhandige primitieve commando's en met een command line interface in plaats van een grafische interface. De reden hiertoe is dat deze oude techniek erg snel is in vergelijking met een operating system met een grafische interface zoals de Microsoft Windows Server. Doordat de Microsoft Windows Server de processor van de server voor een groot deel belast met berekeningen voor de grafische interface, de Active Directory, print server, DNS server, DHCP server en andere processen blijft er minder processor capaciteit over voor het database management system. Dit terwijl de performance (responsetijd) voor de database server juist erg belangrijk is.

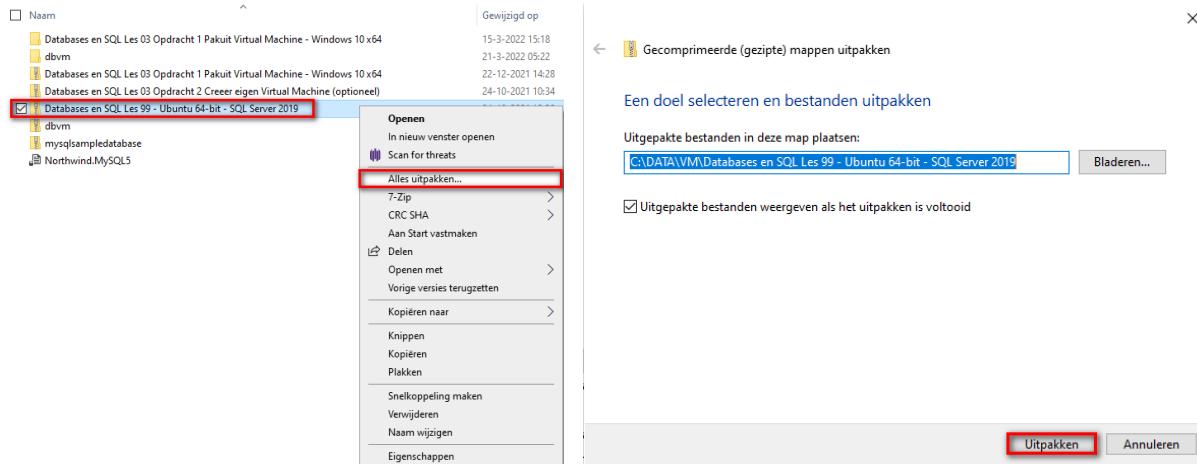
Een Database Management System op Linux of ander UNIX variant is op eenzelfde processor en met dezelfde hoeveelheid geheugen aanzienlijk sneller dan op een Windows server. Dit komt doordat UNIX ontwikkeld was in een tijd dat de processoren aanzienlijk minder krachtig waren dan de huidige processoren en er in die tijd efficiënt met de processoren moest worden omgegaan. Het UNIX operating system belast de processor en het geheugen minimaal zodat de processor nog veel kracht over houdt voor andere taken zoals een database server.

Microsoft Windows Server is een moderner operating system sinds 2000 als opvolger van Microsoft Windows NT Server en is samen met de Intel processor in de laatste decennia sterk in populariteit gegroeid. Door de grafische interface en de grafische tools is Windows Server gemakkelijker te beheren. In plaats van onhandige commando's kunnen met grafische tools direct sprekende overzichten worden getoond van de configuratie of performance van het systeem. Nadeel is echter dat al deze grafische tools ook processorcapaciteit gebruikt en er minder efficiënt met de processoren wordt omgegaan dan een Linux systeem.

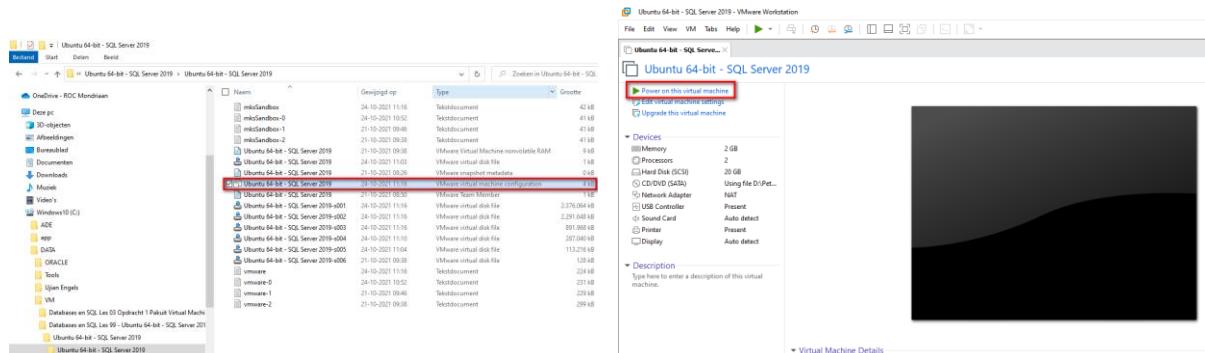
Daarom wordt in de praktijk nog erg veel gebruik gemaakt van Linux of andere UNIX varianten en is het voor een beheerder belangrijk om een beetje wegwijs te zijn in databases op Linux. In dit hoofdstuk zal kennis worden gemaakt met SQL Server op Ubuntu Linux.

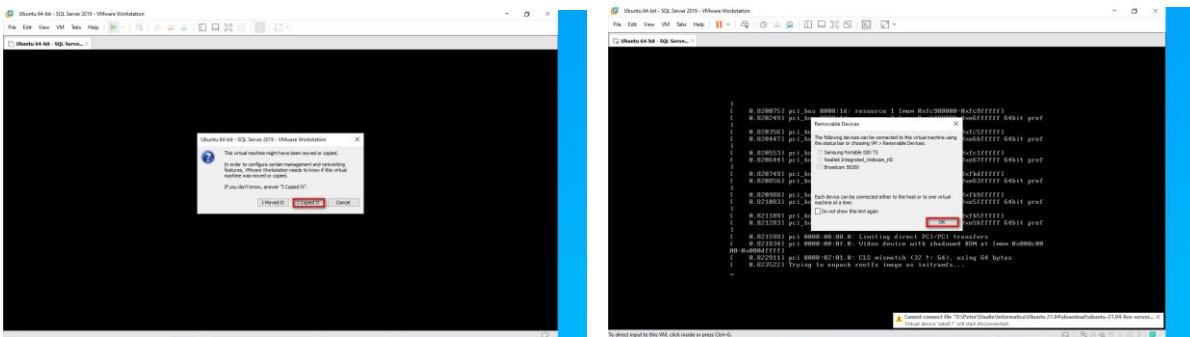
## 28.1 Verbinding maken met SQL Server database op Linux

Pak de zipfile met SQL Server op Ubuntu Virtual Machine uit m.b.v. **Alles uitpakken...**:



Open de Virtual Machine m.b.v. **Power On This Virtual Machine**:





Logon als de Linux gebruiker **roc** en paswoord **Mondriaan**.

```
Ubuntu 21.04 sqlserver2019 tty1
sqlserver2019 login: [ 21.531683] hub 2-2:1.0: hub_ext_port_status failed (err = -110)
sqlserver2019 login: roc
Password: [REDACTED]
```

```
Ubuntu 21.04 sqlserver2019 tty1
sqlserver2019 login: [ 21.531683] hub 2-2:1.0: hub_ext_port_status failed (err = -110)
sqlserver2019 login: roc
Password: [REDACTED]
```

Verander van gebruiker naar de gebruiker root m.b.v. **sudo su -** en geef het paswoord **Mondriaan**.

```
Ubuntu 21.04 sqlserver2019 tty1
sqlserver2019 login: [ 21.531683] hub 2-2:1.0: hub_ext_port_status failed (err = -110)
sqlserver2019 login: roc
Password: Welcome to Ubuntu 21.04 (GNU/Linux 5.11.0-38-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

 System information as of Sun Jun 5 10:39:38 AM UTC 2022

 System load: 0.09      Processes: 248
 Usage of /: 34.4% of 18.57GB  Users logged in: 0
 Memory usage: 45%          IPv4 address for ens3: 192.168.136.130
 Swap usage: 0%

 0 updates can be applied immediately.

 The list of available updates is more than a week old.
 To check for new updates run: sudo apt update

 Last login: Sun Oct 24 09:04:49 UTC 2021 on ttys1
 roc@sqlserver2019:~$ [REDACTED]
roc@sqlserver2019:~$ sudo su -
[sudo] password for roc: [REDACTED]
```

```
Ubuntu 21.04 sqlserver2019 tty1
sqlserver2019 login: [ 21.531683] hub 2-2:1.0: hub_ext_port_status failed (err = -110)
sqlserver2019 login: roc
Password: Welcome to Ubuntu 21.04 (GNU/Linux 5.11.0-38-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

 System information as of Sun Jun 5 10:39:38 AM UTC 2022

 System load: 0.09      Processes: 248
 Usage of /: 34.4% of 18.57GB  Users logged in: 0
 Memory usage: 45%          IPv4 address for ens3: 192.168.136.130
 Swap usage: 0%

 0 updates can be applied immediately.

 The list of available updates is more than a week old.
 To check for new updates run: sudo apt update

 Last login: Sun Oct 24 09:04:49 UTC 2021 on ttys1
 roc@sqlserver2019:~$ [REDACTED]
roc@sqlserver2019:~$ sudo su -
[sudo] password for roc: [REDACTED]
```

Logon op SQL Server m.b.v. **sqlcmd -U SA -P 'Mondriaan'** en selecteer de namen van de aanwezige databases m.b.v. **SELECT Name FROM sys.Databases;** {ENTER} en **GO {ENTER}**

```
root@sqlserver2019:~# sqlcmd -U SA -P 'Mondriaan'
```

```
root@sqlserver2019:~# SELECT Name FROM sys.Databases;
1> SELECT Name FROM sys.Databases;
2> GO
Name
-----
master
tempdb
model
msdb
(4 rows affected)
```

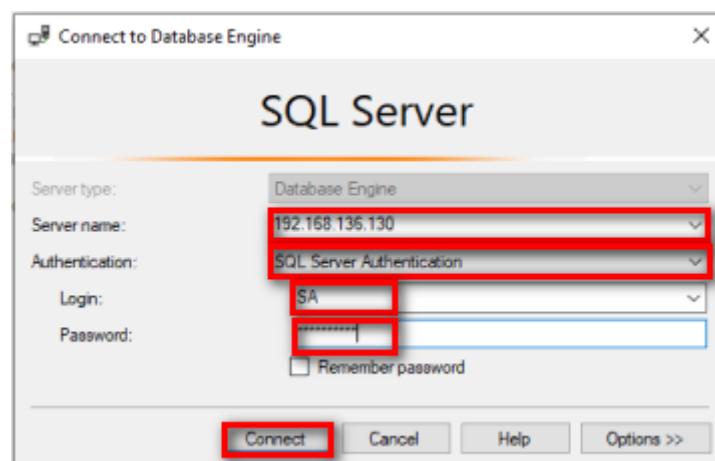
Je bent nu via de command line interface ingelogd in SQL Server RDBMS op de Ubuntu Linux Database server en kan nu alle gewenste SQL commando's invoeren. Met deze command line interface heb je géén grafische tools zoals SSMS op de PC. Het is echter wel mogelijk om vanuit SSMS op de PC of op een Windows Virtual machine verbinding te maken met deze SQL Server op de Ubuntu Linux database server.

Zoek eerst naar het IP-adres van de Ubuntu Linux Server m.b.v. [ip addr](#) en zoek naar het ip adres:

```
root@sqlserver2019:~# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host
            valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:3d:5b:b8 brd ff:ff:ff:ff:ff:ff
    altnet ens33
    inet 192.168.136.130/24 brd 192.168.136.255 scope global dynamic ens33
        valid_lft 1334sec preferred_lft 1334sec
        inet6 fe80::20c:29ff:fe3d:5bb8/64 scope link
            valid_lft forever preferred_lft forever
root@sqlserver2019:~# _
```

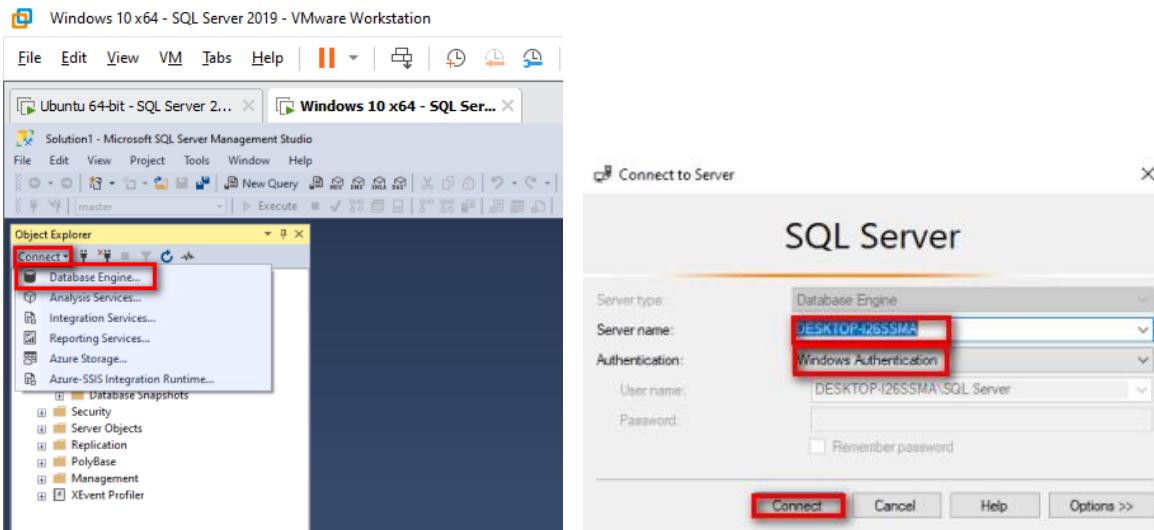
Open SSMS op de Windows Virtual Machine of start SSMS op de PC zelf, voer de onderstaande gegevens in en klik op Connect:

- Server name = <[ip adres uit bovenstaande ip addr](#)>
- Authentication = [SQL Server Authentication](#)
- Login = [SA](#)
- Password = [Mondriaan!](#)

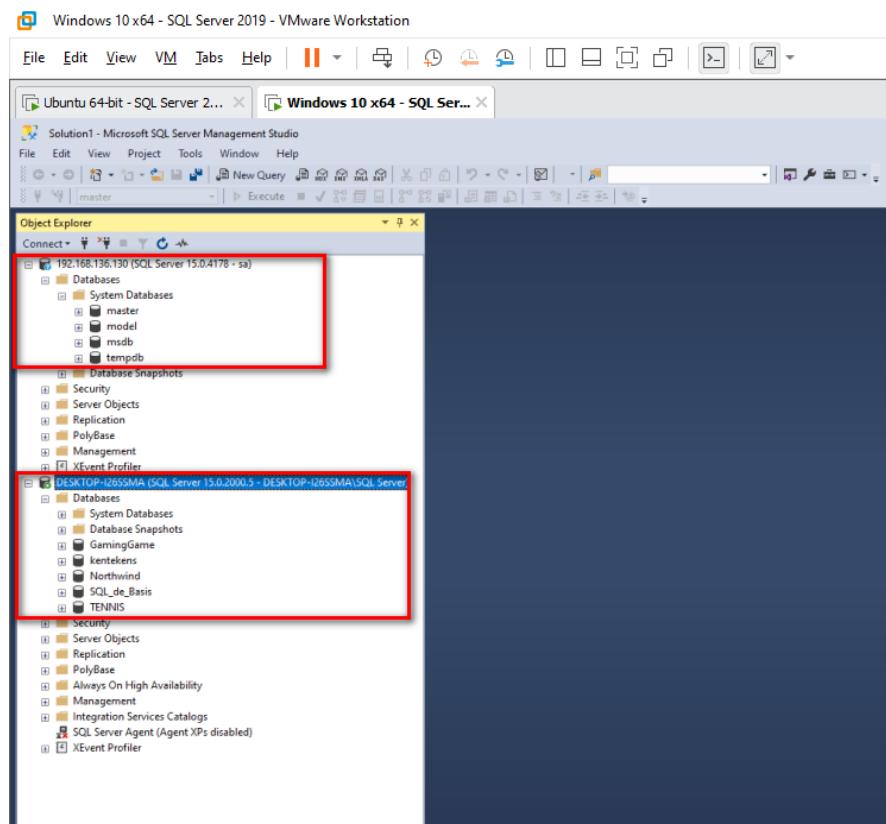


Je bent nu via SSMS ingelogd op de SQL Server RDBMS op de Ubuntu Linux database server. Je kan nu m.b.v. de SSMS Graphical User Interface alle acties uitvoeren die je ook op de lokale SQL Server

RDBMS kan uitvoeren. De Grafical User Interface gebruikt nu voor de grafische schermen de processor capaciteit van de PC zelf en niet de processor capaciteit van de Ubuntu Linux database server (Client-Server concept).



Door op de Connect knop te klikken kan je nu ook nog een verbinding maken met de lokale SQL Server RDBMS. Je bent dan ingelogd op de SQL Server RDBMS op de Ubuntu Linux Database Server en op de Lokale SQL Server RDBMS:



Het is indien gewenst vanuit de PC of Virtual Machine ook mogelijk om een aparte ssh verbinding te maken met de Ubuntu Linux server m.b.v. <ssh roc@<ip adres van de Ubuntu Linux server>>.

## 28.2 Wegwijs in SQL Server op Linux

De SQL Server RDBMS configuratie kan onder root worden aangepast m.b.v. [/opt/mssql/bin/mssql-opt](#) nadat het RDBMS is gestopt.

Het configuratiebestand van SQL Server is [/var/opt/mssql/mssql.conf](#)

SQL Server RDBMS kan onder root worden gestopt m.b.v. [systemctl stop mssql-server](#)

SQL Server RDBMS kan onder root worden opgestart m.b.v. [systemctl start mssql-server](#)

Inloggen in SQL Server command line interface kan onder root m.b.v. [sqlcmd -U SA -P 'Mondriaan!'](#)

De default locatie voor de database files en transaction log files is [/var/opt/mssql/data](#)

De default locatie voor de error logfiles is [/var/opt/mssql/log](#)

De default locatie voor de backup bestanden (.bak) is [/var/opt/mssql/data](#)

Met het commando [ps -ef | grep sql](#) kan de processen van SQL Server zichtbaar worden gemaakt:

```
root@sqlserver2019:~# ps -ef | grep sql
mssql      885      1  3 12:48 ?        00:00:04 /opt/mssql/bin/sqlservr
mssql     1098     885  5 12:48 ?        00:00:06 /opt/mssql/bin/sqlservr
root      1449    1393  0 12:50 pts/1      00:00:00 grep sql
root@sqlserver2019:~#
```

Met het commando [top](#) kan worden gekeken naar de zwaarste processen en het geheugengebruik zichtbaar worden gemaakt:

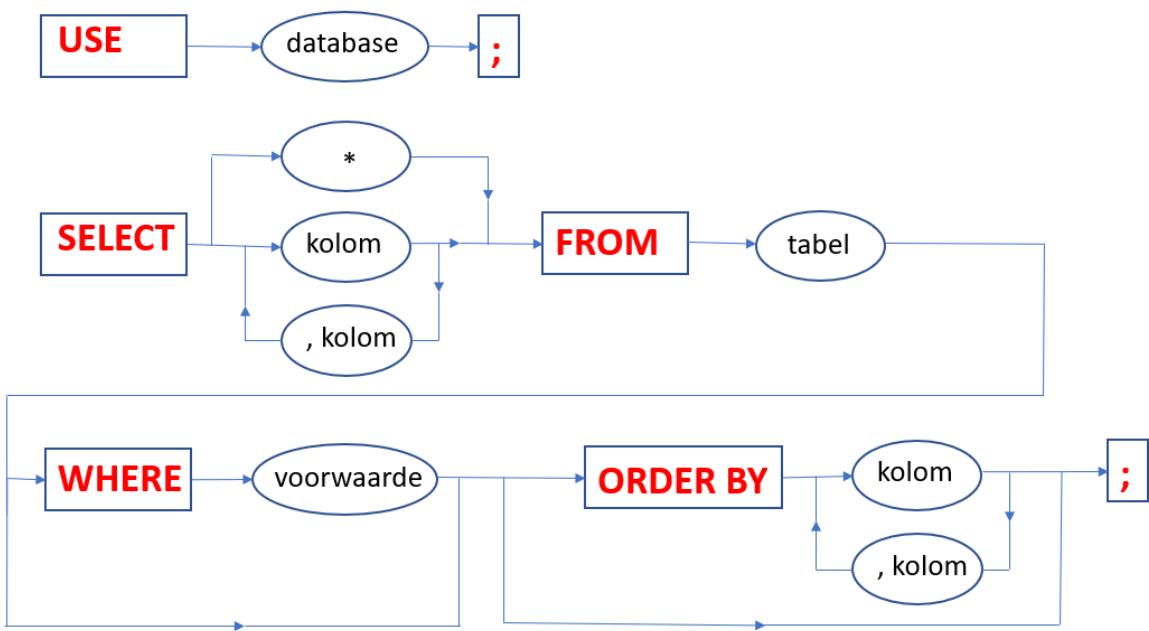
```
top - 13:47:11 up 58 min,  2 users,  load average: 0.07, 0.07, 0.07
Tasks: 225 total,   1 running, 224 sleeping,   0 stopped,   0 zombie
%CPU(s):  0.2 us,  0.2 sy,  0.0 ni, 99.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem : 1946.3 total,   68.0 free, 1052.2 used,   826.0 buff/cache
MiB Swap: 2048.0 total,  2033.5 free,    14.5 used.   739.0 avail Mem

      PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM TIME+ COMMAND
 2546 mssql    20   0 6371304 790396 58232 S  1.7 39.7  0:44.39 sqlservr
  500 root    19  -1  56536 16720 15740 S  0.3  0.8  0:01.32 systemd-journal
 2979 root    20   0     0     0   0 I  0.3  0.0  0:01.58 kworker/1:1-events
 3350 roc    20   0 15128  6292   4796 S  0.3  0.3  0:00.08 sshd
 3874 root    20   0 10340  4228   3436 R  0.3  0.2  0:00.06 top
     1 root    20   0 166588 10488  7732 S  0.0  0.5  0:02.30 systemd
     2 root    20   0     0     0   0 S  0.0  0.0  0:00.01 kthreadd
     3 root    0  -20     0     0   0 I  0.0  0.0  0:00.00 rcu_gp
     4 root    0  -20     0     0   0 I  0.0  0.0  0:00.00 rcu_par_gp
     6 root    0  -20     0     0   0 I  0.0  0.0  0:00.00 kworker/0:0H-events_highpri
     9 root    0  -20     0     0   0 I  0.0  0.0  0:00.00 mm_percpu_wq
    10 root    20   0     0     0   0 S  0.0  0.0  0:00.00 rcu_tasks_rude_
    11 root    20   0     0     0   0 S  0.0  0.0  0:00.00 rcu_tasks_trace
    12 root    20   0     0     0   0 S  0.0  0.0  0:00.06 ksoftirqd/0
    13 root    20   0     0     0   0 I  0.0  0.0  0:00.91 rcu_sched
```

## Bijlage 1 Syntax Diagrams

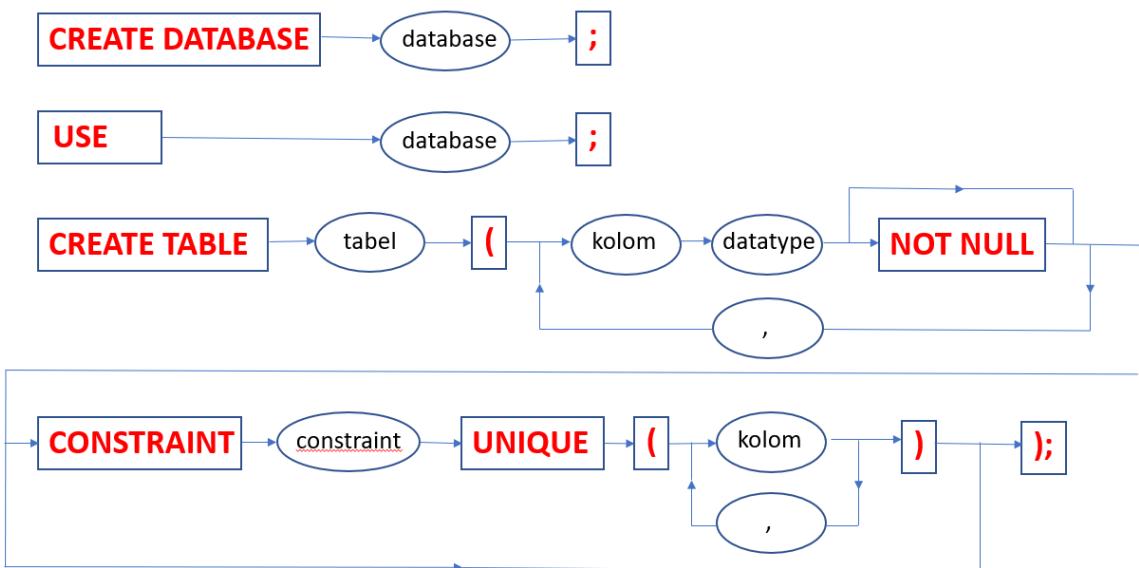
### SELECT

# SELECT

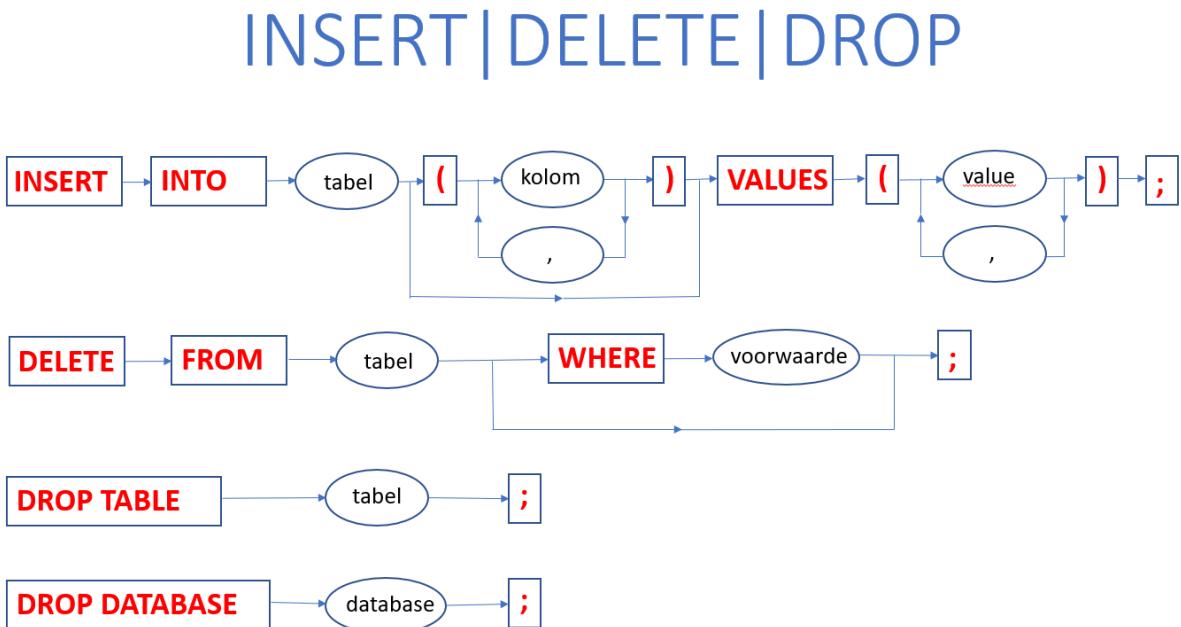


### CREATE

## CREATE DATABASE | TABLE

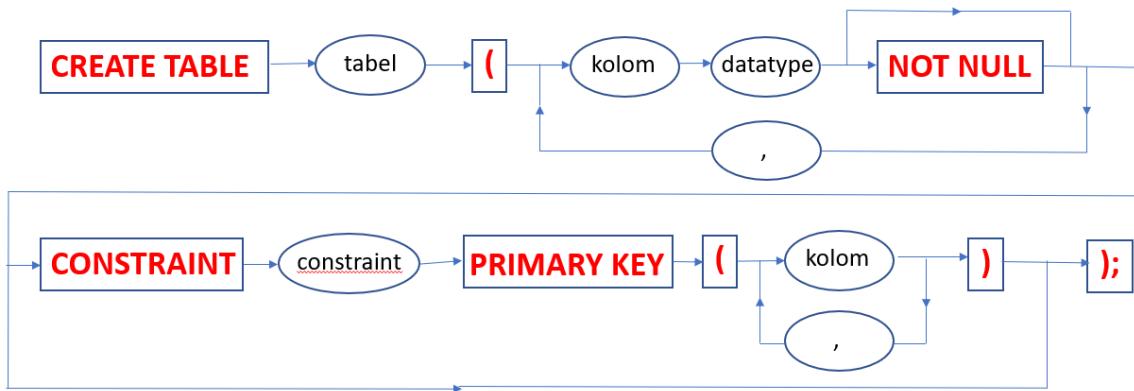


### INSERT, DELETE, DROP



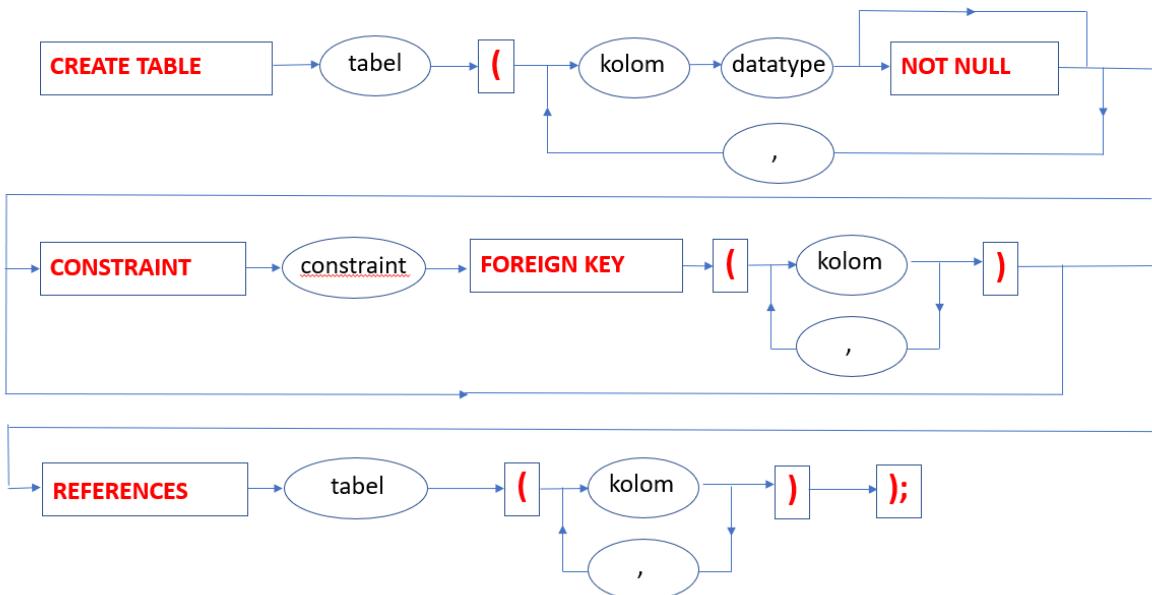
### PRIMARY KEY Constraint

## Primary Key constraint

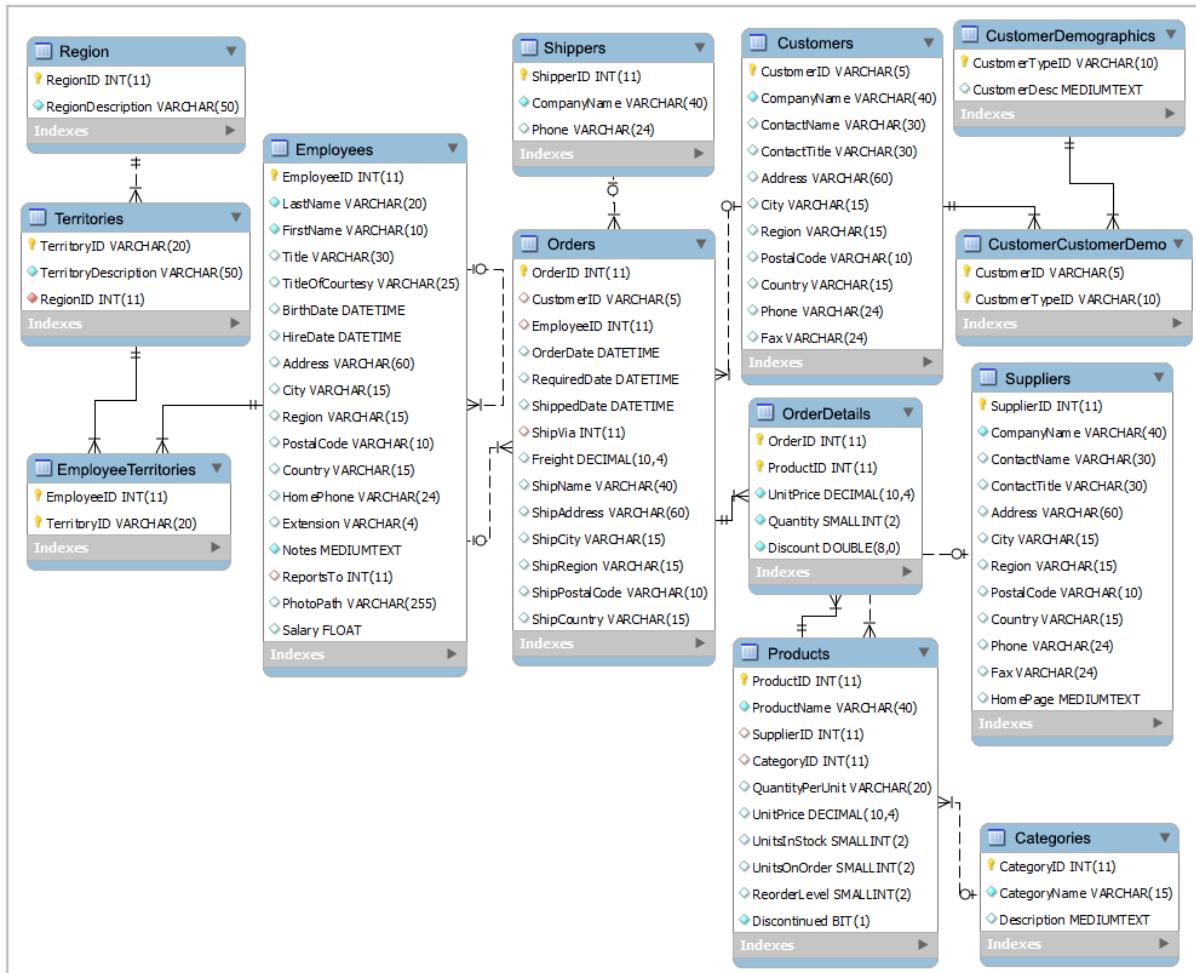


## FOREIGN KEY Constraint

## FOREIGN KEY constraint



## Bijlage 2 Entity Relation Diagram van Northwind database



### Bijlage 3 SQL script voor genereren Northwind Trades Factuur

```

PRINT 'Northwind Traders';
PRINT '3066 Long Rd, Green Lane';
PRINT 'Pennsylvania';
PRINT '18054';
PRINT 'United States';
PRINT '';

DECLARE @OrderID INT = 11071;
DECLARE @CompanyName NVARCHAR(20);
SELECT @CompanyName = c.CompanyName FROM Orders AS o, Customers AS c
WHERE o.CustomerID = c.CustomerID
AND OrderID = @OrderID;
PRINT '                                ' + @CompanyName;

DECLARE @ContactName NVARCHAR(20);
SELECT @ContactName = c.ContactName FROM Orders AS o, Customers AS c
WHERE o.CustomerID = c.CustomerID
AND OrderID = @OrderID;
PRINT '                                attn. ' + @ContactName;

DECLARE @Address NVARCHAR(20);
SELECT @Address = c.Address FROM Orders AS o, Customers AS c
WHERE o.CustomerID = c.CustomerID
AND OrderID = @OrderID;
PRINT '                                ' + @Address;

DECLARE @City NVARCHAR(20);
SELECT @City = c.City FROM Orders AS o, Customers AS c
WHERE o.CustomerID = c.CustomerID
AND OrderID = @OrderID;
PRINT '                                ' + @City;

DECLARE @PostalCode NVARCHAR(20);
SELECT @PostalCode = c.PostalCode FROM Orders AS o, Customers AS c
WHERE o.CustomerID = c.CustomerID
AND OrderID = @OrderID;
PRINT '                                ' + @PostalCode;

DECLARE @Country NVARCHAR(20);
SELECT @Country = c.Country FROM Orders AS o, Customers AS c
WHERE o.CustomerID = c.CustomerID
AND OrderID = @OrderID;
PRINT '                                ' + @Country;
PRINT '';

DECLARE @Sysdate NVARCHAR(20);
SELECT @Sysdate = CONVERT (date, SYSDATETIME()) ;

PRINT 'Pennsylvania' + ',' + @Sysdate
PRINT '';

PRINT 'INVOICE'
PRINT 'OrderID      : ' + CONVERT(VARCHAR(10),@OrderID);
DECLARE @EmployeeID INT;
DECLARE @EmployeeLastName NVARCHAR(20);
DECLARE @EmployeeFirstName NVARCHAR(20);
SELECT @EmployeeID = EmployeeID FROM Orders WHERE OrderID = @OrderID;

```

```

SELECT @EmployeeLastName = e.LastName FROM Orders o, Employees e WHERE o.EmployeeID
= e.EmployeeID AND OrderID = @OrderID;
SELECT @EmployeeFirstName = e.FirstName FROM Orders o, Employees e WHERE
o.EmployeeID = e.EmployeeID AND OrderID = @OrderID;

PRINT 'Sales Employee      : ' + @EmployeeFirstName + ' ' + @EmployeeLastName;
PRINT '';
PRINT '';
PRINT '';

DECLARE @ProductID INT;
SELECT @ProductID = d.ProductID FROM [Order Details] as d;
DECLARE @ProductName NVARCHAR(40);
SELECT @ProductName = p.ProductName FROM Products as p;
DECLARE @Quantity INT;
SELECT @Quantity = d.Quantity FROM [Order Details] as d;
DECLARE @Discount INT;
SELECT @Discount = d.Discount FROM [Order Details] as d;

SET NOCOUNT ON;

SELECT d.ProductID, p.ProductName,d.UnitPrice, d.Quantity, d.Discount, (d.UnitPrice
* d.Quantity * (1 - Discount)) AS Cost FROM [Order Details] as d, Products as p
WHERE d.ProductID = p.ProductID AND d.OrderID = @OrderID
GROUP BY d.ProductID, p.ProductName,d.UnitPrice, d.Quantity, d.Discount;

WITH CTE_ORDERS AS
(
SELECT [OrderID],[ProductID],[UnitPrice],[Quantity],[Discount],
SUM([UnitPrice] * [Quantity]) * (1 - Discount) AS Total
FROM [dbo].[Order Details] WHERE OrderID = 11071
GROUP BY [OrderID],[ProductID],[UnitPrice],[Quantity],[Discount]
)
SELECT '
',SUM(Total) AS GrandTotal FROM CTE_ORDERS;

```