

Gevorderd dynamische website

Plannen en ontwerpen

hoofdstuk

2

MSSQL



KONING
WILLEM I
COLLEGE



Algemene informatie

Onderwerp	Kennismaken met SQL-Server en SQL management studio.
Leerdoel(en)	<ol style="list-style-type: none">1. De student is in staat om voor attributen een geschikt datatype te kiezen die aansluit bij de informatie uit de datadictionary.2. De student is in staat om door middel van SQL query's een CREATE-script op te stellen waarbij databases, tabellen, attributen, primary keys, alternate keys en foreign keys zijn opgenomen.3. De student is in staat om vanuit SQL-management studio een ERD te genereren zodat hij kan controleren of de aangemaakte database overeenkomt met een ontwerp.
Vereiste voorkennis	De student moet in staat zijn om een datadictionary en ERD te kunnen lezen. Wat betreft SQL hoeft de student geen specifieke voorkennis te hebben.
Kwalificatiedossier	<ul style="list-style-type: none"><input type="checkbox"/> B1-K1-W1: Plant werkzaamheden en bewaakt de voortgang<input checked="" type="checkbox"/> B1-K1-W2: Ontwerpt software<input checked="" type="checkbox"/> B1-K1-W3: Realiseert (onderdelen van) software<input type="checkbox"/> B1-K1-W4: Test software<input type="checkbox"/> B1-K1-W5: Doet verbetervoorstellen voor de software <input type="checkbox"/> B1-K2-W1: Voert overleg<input type="checkbox"/> B1-K2-W2: Presenteert het opgeleverde werk<input type="checkbox"/> B1-K2-W3: Reflecteert op het werk



Inhoudsopgave

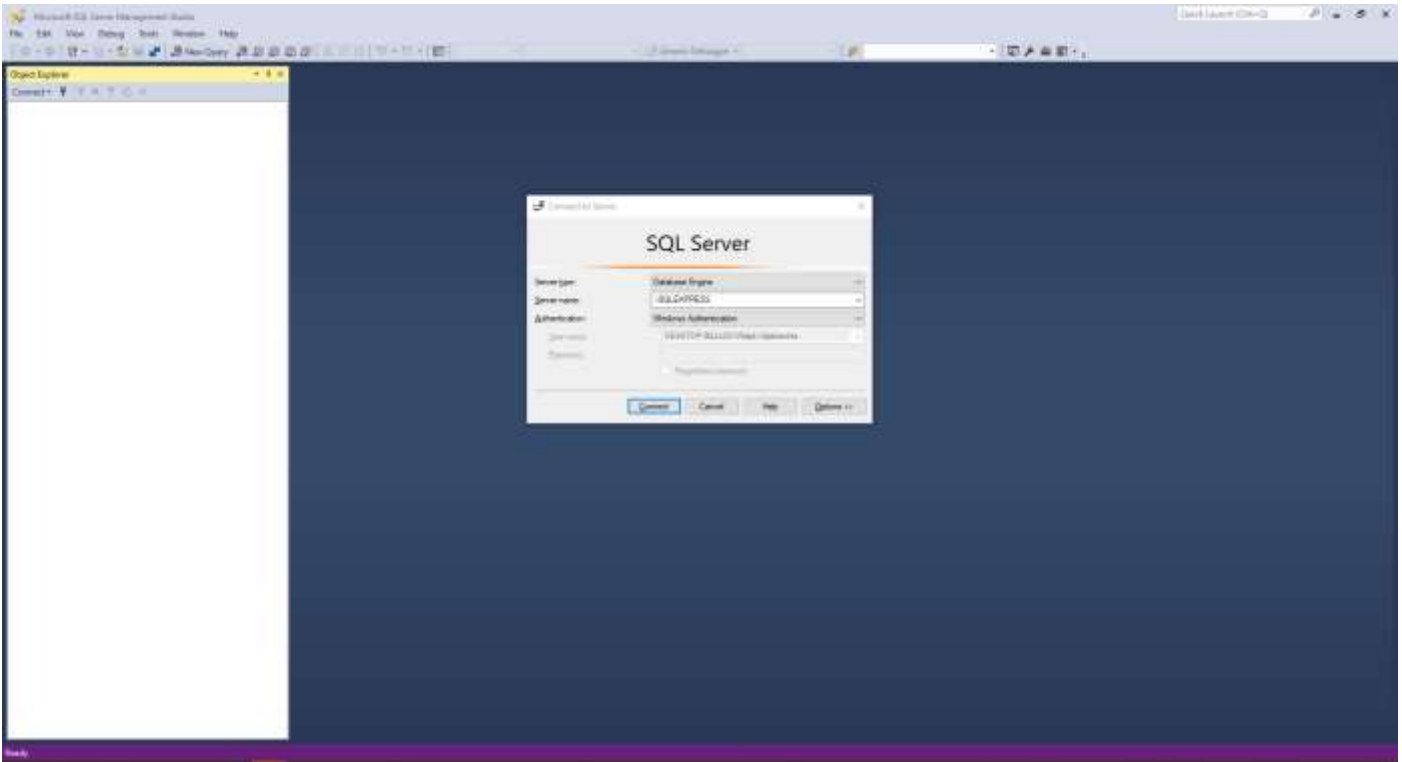
Algemene informatie	2
Inhoudsopgave	3
Inhoud	4
Database Management System (DBMS)	4
Structured Query Language (SQL)	4
Ontstaan van SQL	4
Database en SQL.....	5
De opbouw van een CREATE-script.....	6
Het maken van een database	7
Code uitvoeren	7
Het maken van een tabel	7
Datatypen.....	9
Niet leeg.....	10
Automatische nummering	10
Sleutels.....	11
Primary keys	11
Alternate keys.....	11
Foreign keys	12
ERD genereren vanuit SQL Management Studio.....	12



Inhoud

Database Management System (DBMS)

Een DBMS is een systeem, in dit geval een applicatie, waarmee je databases kunt bouwen en de gegevens ervan kunt beheren. De DBMS die je hier op school gebruikt heet SQL Management Studio. In de eerste oefening van dit hoofdstuk ga je deze software installeren op basis van een installatiehandleiding.



Structured Query Language (SQL)

Ontstaan 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 datasystemen.

Aan de hand van dit relationele model bouwt IBM een experimenteel systeem genaamd 'System R'. Omdat er ook een mogelijkheid moet zijn om data in 'System R' te lezen en te bewerken, ontwikkelen Donald Chamberlin en Raymond Boyce de taal 'Structured English Query Language', of SEQUEL. Helaas zat er aan het woord SEQUEL een geregistreerd handelsmerk van een Engelse vliegtuigbouwer en dus 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 verschijnen, maakt IBM zich in 1978 op om als onderdeel van System/38 het eerste relationele databasesysteem met SQL 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 als bedrijfsnaam).



Database en SQL

Om een database te bouwen in een DBMS of gegevens daarin te beheren maak je gebruik van SQL-opdrachten, ook wel SQL-statements genoemd.

SQL-opdrachten zijn te onderscheiden naar vier typen:

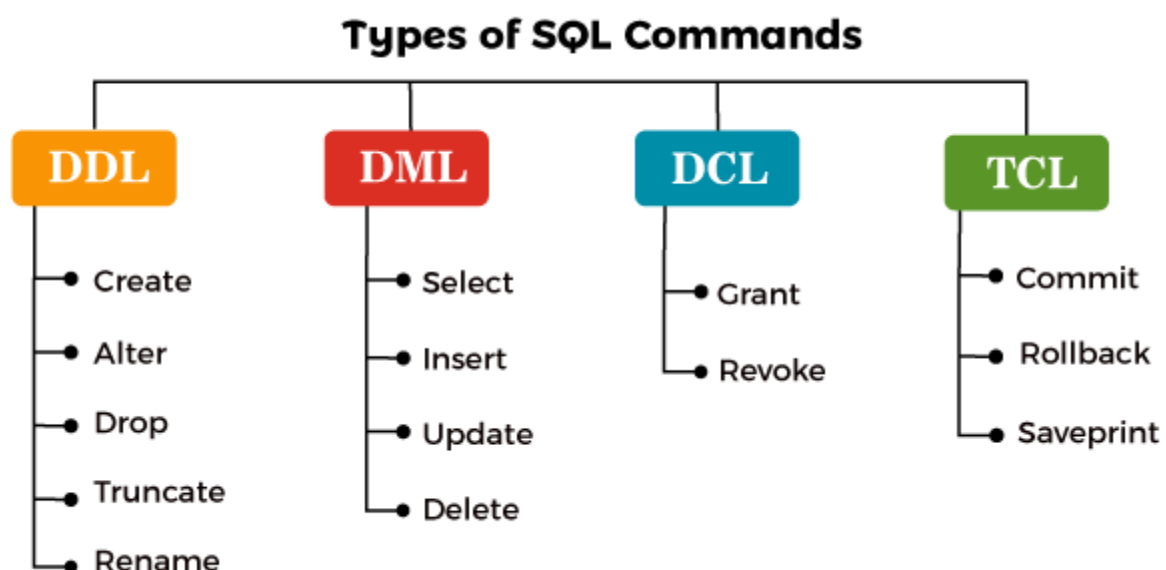
1. **Data Definition Language (DDL)**
2. **Data Control Language (DCL)**
3. **Data Manipulation Language (DML)**
4. **Transaction Control Language (TCL)**

Voor zover er nog geen database beschikbaar is, moet deze worden aangemaakt. Hiertoe kunnen de DDL-statements worden gebruikt. Anders gezegd: met behulp van de DDL-statements kan de structuur van de database worden beheerd. Dit zal dan ook na het maken van een database model bij plannen en ontwerpen worden geleerd.

Is er eenmaal een database beschikbaar, dan moet je kunnen bepalen wie er met de ingevoerde gegevens moet kunnen werken. Er moeten rechten kunnen worden toegekend aan gebruikers. Het beheer van de rechten wordt geregeld met behulp van de DCL-statements. Dit onderdeel zal terugkomen bij de module computervaardigheden.

DML-statements worden over het algemeen door gebruikers toegepast. Zij selecteren gegevens op basis van de bestaande informatiebehoefte, voeren nieuwe gegevens in, wijzigen en verwijderen gegevens. Dit onderdeel zal terugkomen bij de module realiseren. Om goede programmacode te kunnen schrijven zal je ook de verschillende DML-statements moeten kennen.

TCL-statements komen in je opleiding verder niet aan bod. Dit zijn meer de statements die passen bij taken die een beheerder uitvoert.





De opbouw van een CREATE-script

Een CREATE-script is alle code die nodig is om de database aan te maken. Daarbij maak je altijd gebruik van onderstaande opbouw. Onderstaande commentaarregels dien je altijd minimaal te gebruiken in je CREATE-script.

Commentaar schrijf je in SQL met "/* */" voor meerdere regels, of met "--" per regel.

```
1  /*
2      Auteur:          ...
3      Aanmaakdatum:   ...
4      Omschrijving:    CREATE-script ...
5  */
6
7  -- Database aanmaken
8
9  -- Tabellen aanmaken
10
11 -- Primary keys aanmaken
12
13 -- Alternate keys aanmaken
14
15 -- Foreign keys aanmaken
16
17 -- Data toevoegen
18     -- Laag 1
19     -- Laag 2
20     -- Etc.
```



Het maken van een database

Voordat je een tabel kunt aanmaken moet je eerst de database zelf aanmaken. Een database maak je aan in de master database. Zie het als een database die bijhoudt welke databases er op de server draaien. De master database houdt nog veel meer bij, maar dat is voor nu niet belangrijk om te weten.


```
7  -- Database aanmaken
8  USE master;
9  DROP DATABASE IF EXISTS voorbeeld;
10 CREATE DATABASE voorbeeld;
11 GO
12 USE voorbeeld;
```

Uitleg code:

Regel	Uitleg
8	Aangeven dat je de master database wilt gaan gebruiken.
9	De database verwijderen indien de database al bestaat.
10	De database aanmaken.
11	Een GO zorgt er voor dat de code tot aan de GO gezien wordt als één geheel (batch). De code die ernaar komt, wordt pas uitgevoerd als alle code tot de GO is uitgevoerd.
12	

Achter een GO zet je nooit een ";".

Code uitvoeren

Door in de menubalk op  **Execute** te klikken kun je de code uitvoeren. Als je geen code geselecteerd hebt, dan wordt standaard de gehele code uitgevoerd. Daarbij moet je ook altijd goed kijken in welke database de code wordt uitgevoerd. Dit kun je aangeven in een dropdownlist die in het menu staat.



Je kunt dit echter ook vanuit de code zelf doen. Dit zie je gebeuren op regel 8 in het voorbeeld. Dat is heel handig! Want nadat je een database gemaakt hebt, wil je in die database de tabellen, attributen en sleutels gaan plaatsen.

Het maken van een tabel

Nadat je de database hebt aangemaakt ga je eerst alle tabellen met de daarbij horende attributen aanmaken. Pas later ga je de sleutels toevoegen. Dit zie je ook terug in de opbouw van een CREATE-script. Maak de rest van je CREATE-script aan op basis van je datadictionary. Dat is hetgeen waar de meest gedetailleerde informatie te vinden is voor het bouwen van je database.



Datadictionary:

Diagram 1				Patient		
Sleutels			Attribuutnaam	Datatype	Lengte	Voorwaarde invulwaarde NL / OPT / AUTO(x,x)
PK	AK	FK				
PK1			Patientnummer	Numeriek	6	Niet leeg, AUTO(100000,1)
			Aanhef	Karakters	4	Niet leeg
			Voorletters	Karakters	5	Niet leeg
			Achternaam	Karakters	40	Niet leeg
	A1		Emailadres	Karakters	50	Niet leeg
			Geboortedatum	Datum		Niet leeg
	A2		Telefoonnummer	Karakters	10	Optioneel

Omgezet naar SQL-code:

```
14 -- Tabellen aanmaken
15 CREATE TABLE Patient
16 (
17
18
19
20
21
22
23
24 );
```

Patientnummer	INT	NOT NULL	IDENTITY(100000,1),
Aanhef	VARCHAR(4)	NOT NULL,	
Voorletter	CHAR(1)	NOT NULL,	
Achternaam	VARCHAR(40)	NOT NULL,	
Emailadres	VARCHAR(50)	NOT NULL,	
Geboortedatum	DATE	NOT NULL,	
Telefoonnummer	VARCHAR(15)		

Uitleg code:

Regel	Uitleg
14	Aanmaken van een tabel.
15	Voordat je de attributen gaat opnemen die in de tabel zitten plaats je een "(".
16-22	Aanmaken van de attributen die in de tabel horen. Het einde van de regel sluit je af met een "," behalve bij het laatste attribuut.
23	Nadat je alle attributen gemaakt hebt plaats je een ");". Let op, inclusief ";"!.



Datatypes

Bij het aanmaken van een attribuut dien je ook altijd een datatype mee te geven. In je datadictionary heb je daar al over nagedacht. Echter is dat heel algemeen. Je kunt een database maken in verschillende talen (zoals MySQL en MSSQL). De datatypes die verschillen per taal. Nu dien je de vertaling te maken van de algemene datatypes naar de specifieke datatypes van de taal die je gebruikt om de database te maken. In dit geval is dat MSSQL. Hieronder volgt een overzicht van de datatypes die je kunt gebruiken:

Karakters

Data type	Description	Max size	Storage
char(n)	Fixed width character string	8,000 characters	Defined width
varchar(n)	Variable width character string	8,000 characters	2 bytes + number of chars
varchar(max)	Variable width character string	1,073,741,824 characters	2 bytes + number of chars
text	Variable width character string	2GB of text data	4 bytes + number of chars
nchar	Fixed width Unicode string	4,000 characters	Defined width x 2
nvarchar	Variable width Unicode string	4,000 characters	
nvarchar(max)	Variable width Unicode string	536,870,912 characters	
ntext	Variable width Unicode string	2GB of text data	
binary(n)	Fixed width binary string	8,000 bytes	
varbinary	Variable width binary string	8,000 bytes	
varbinary(max)	Variable width binary string	2GB	
image	Variable width binary string	2GB	

Numeriek en decimaal

Data type	Description	Storage
bit	Integer that can be 0, 1, or NULL	
tinyint	Allows whole numbers from 0 to 255	1 byte
smallint	Allows whole numbers between -32,768 and 32,767	2 bytes
int	Allows whole numbers between -2,147,483,648 and 2,147,483,647	4 bytes
bigint	Allows whole numbers between -9,223,372,036,854,775,808 and 9,223,372,036,854,775,807	8 bytes
decimal(p,s)	Fixed precision and scale numbers. Allows numbers from $-10^{38} + 1$ to $10^{38} - 1$. The p parameter indicates the maximum total number of digits that can be stored (both to the left and to the right of the decimal point). p must be a value from 1 to 38. Default is 18. The s parameter indicates the maximum number of digits stored to the right of the decimal point. s must be a value from 0 to p. Default value is 0	5-17 bytes



numeric(p,s)	Fixed precision and scale numbers. Allows numbers from $-10^{38} + 1$ to $10^{38} - 1$. The p parameter indicates the maximum total number of digits that can be stored (both to the left and to the right of the decimal point). p must be a value from 1 to 38. Default is 18. The s parameter indicates the maximum number of digits stored to the right of the decimal point. s must be a value from 0 to p. Default value is 0.	5-17 bytes
smallmoney	Monetary data from -214,748.3648 to 214,748.3647	4 bytes
money	Monetary data from -922,337,203,685,477.5808 to 922,337,203,685,477.5807	8 bytes
float(n)	Floating precision number data from $-1.79E + 308$ to $1.79E + 308$. The n parameter indicates whether the field should hold 4 or 8 bytes. float(24) holds a 4-byte field and float(53) holds an 8-byte field. Default value of n is 53.	4 or 8 bytes
real	Floating precision number data from $-3.40E + 38$ to $3.40E + 38$	4 bytes

Datum en tijd

Data type	Description	Storage
datetime	From January 1, 1753 to December 31, 9999 with an accuracy of 3.33 milliseconds	8 bytes
datetime2	From January 1, 0001 to December 31, 9999 with an accuracy of 100 nanoseconds	6-8 bytes
smalldatetime	From January 1, 1900 to June 6, 2079 with an accuracy of 1 minute	4 bytes
date	Store a date only. From January 1, 0001 to December 31, 9999	3 bytes
time	Store a time only to an accuracy of 100 nanoseconds	3-5 bytes

Niet leeg

Met een NOT NULL achter het datatype van het attribuut geef je aan dat de waarde verplicht moet worden ingevuld zodra je gegevens aan de tabel gaat toevoegen. Als je dit niet opgeeft, betekent dit automatisch dat het optioneel is.

18 | Aanhef VARCHAR(4) NOT NULL,

Automatische nummering

Met een IDENTITY kun je de waarde van het attribuut automatisch laten nummeren. Achter de identity zet je tussen de ronde haken twee getallen gescheiden met een komma. Het eerste getal is het startgetal. Het tweede getal is om aan te geven met hoeveel de waarde steeds opgehoogd dient te worden als de record (object) aangemaakt wordt.

17 | Patientnummer INT NOT NULL IDENTITY(100000,1),

In dit voorbeeld krijgt de eerste patient die wordt aangemaakt een patientnummer 100000, de tweede 100001, de derde 100002 etc.



Sleutels

Nadat je de tabellen hebt aangemaakt ga je de sleutels toevoegen. Gebruik de namen die je beschreven hebt in de datadictionary.

Primary keys

Een primary key maak je aan met onderstaande code:

```
56 | -- Primary keys aanmaken
57 | ALTER TABLE Patient
58 | ADD CONSTRAINT PK_Patient
59 | PRIMARY KEY (Patientnummer);
```

Regel	Uitleg
57	Je past de tabel aan. Dit is dus de tabel waarin je de primary key wilt aanmaken.
58	Je geeft de primary key een naam zoals in de datadictionary.
59	Hier geef je het attribuut of attributen op tussen de ronde haken om aan te geven waar de primary key op ligt.

Voorbeeld met een primary key over meerdere velden:

```
73 | ALTER TABLE AfspraakBehandeling
74 | ADD CONSTRAINT PK_AfspraakBehandeling
75 | PRIMARY KEY (Afspraaknummer, Behandelingscode);
```

Het gaat hier dus om één samengestelde primary key.

Alternate keys

Een alternate key maak je aan met onderstaande code:

```
77 | -- Alternate keys aanmaken
78 | ALTER TABLE Patient
79 | ADD CONSTRAINT AK_Patient_Emailadres
80 | UNIQUE (Emailadres);
```

Regel	Uitleg
78	Je past de tabel aan. Dit is dus de tabel waarin je de alternate key wilt aanmaken.
79	Je geeft de alternate key een naam zoals in de datadictionary.
80	Hier geef je het attribuut of attributen op tussen de ronde haken om aan te geven waar de primary key op ligt.

Voorbeeld met een alternate key over meerdere velden:

```
86 | ALTER TABLE Afspraak
87 | ADD CONSTRAINT AK_Afspraak_StartmomentPersoneelsnummer
88 | UNIQUE (Startmoment, Personeelsnummer);
```

Het gaat hier dus om één samengestelde alternate key.



Foreign keys

Een foreign key maak je aan met onderstaande code:

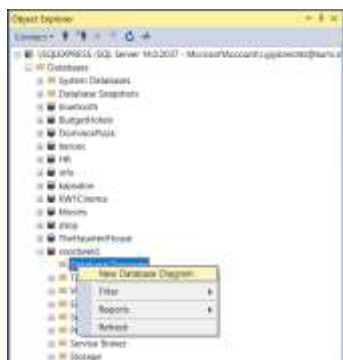
```
102 -- Foreign keys aanmaken
103 ALTER TABLE Afspraak
104 ADD CONSTRAINT FK_Afspraak_Patient
105 FOREIGN KEY (Patientnummer)
106 REFERENCES Patient(Patientnummer);
```

Regel	Uitleg
103	Je past de tabel aan. Dit is dus de tabel waarin je de alternate key wilt aanmaken.
104	Je geeft de alternate key een naam zoals in de datadictionary.
105	Hier geef je het attribuut of attributen op tussen de ronde haken om aan te geven waar de foreign key op ligt.
106	Hier geef je de tabel en de primary key van die tabel op, om aan te geven waar de foreign key naar refereert.

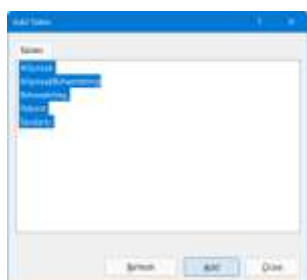
ERD genereren vanuit SQL Management Studio

Nadat je een database hebt aangemaakt kun je een ERD laten genereren vanuit SQL Management Studio. Je kunt dan gemakkelijk zien of de gegenereerde ERD overeenkomt met de ERD die je tijdens het ontwerp gemaakt hebt.

Hiervoor ga je in de map databases op zoek naar de database waarvan je de ERD wilt laten maken. Klap de database open en klik met rechts op Database Diagrams. Kies voor New Database Diagram.



Selecteer in het venster dat opkomt alle tabellen (control toets en één voor één aanklikken) en klik op Add.



Klik daarna op Close om de ERD te kunnen bekijken.

