

leren. durven. doen.



DATABANKEN - SQL

C# programmeur

SQL – INHOUD

- **Wijzigen van data**
 - **Insert - update - delete**
- **DDL**
- **Tabellen en relaties**
- **Views**

DATABANKEN - SQL

Wijzigen van data

Tot nu toe...

SQL = **Data Manipulation Language (DML)** +
Data Definition Language (**DDL**) +
Data Control Language (**DCL**)

DML

SELECT
WHERE
ORDER BY
AGGREGATE FUNCTIONS
GROUP BY
HAVING
JOINS
UNION

Nu...

- **Vervolg Data Manipulation Language**
 - Toevoegen van data
 - Wijzigen van data
 - Verwijderen van data

DML

INSERT

INSERT INTO ... SELECT

UPDATE

DELETE

Rij toevoegen aan tabel: INSERT INTO...

Syntaxis

```
INSERT INTO table_name (column1, column2, ...)  
VALUES (value1, value2, ...),  
      (anothervalue1, anothervalue2, ..), ...
```

```
INSERT INTO Klanten ([RijkRegNr] ,[VoorNaam],[FamilieNaam])  
VALUES ('47089608569', 'Jan','Janssens','jan.janssens@gmail.com'),  
      ('75859674115', 'Piet','Pieters','piet.pieters@hotmail.com');
```

Opmerking

Als je niet elk veld opgeeft, wordt in de ontbrekende kolommen de standaardwaarde of **NULL** ingevoegd.

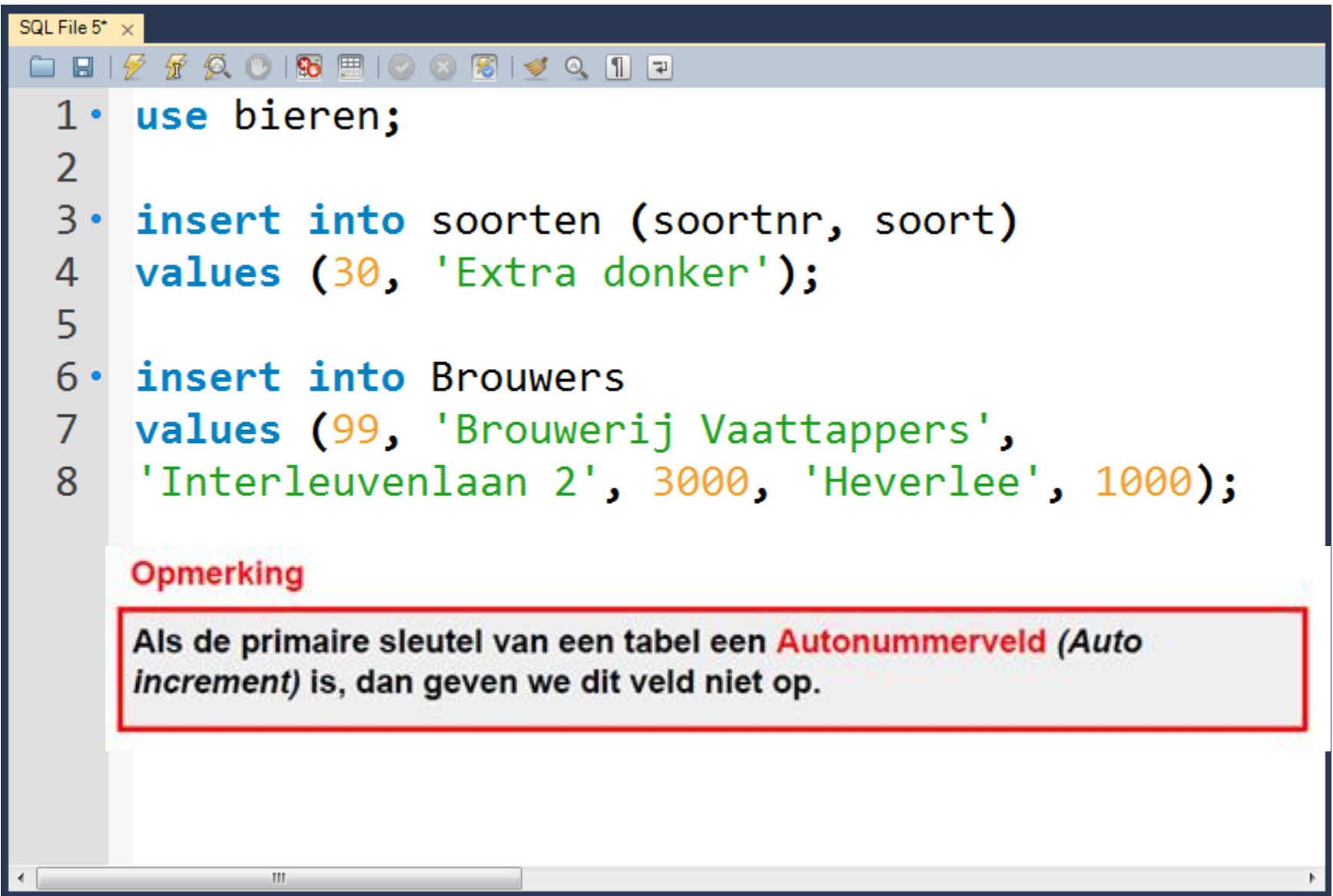
Oplossing vorige opdracht

```
USE plantv;  
INSERT INTO leveran (lev_code, lev_naam)  
VALUES ('453','Pieter');
```

Resultaat

Result Grid	Filter Rows:	Search	Export:
LEV_CODE	LEV_NAAM	ADRES	WOONPL
004	HOVENIER G.H.	ZANDWEG 50	LISSE
009	BAUMGARTEN R.	TAKSTRAAT 13	HILLEGOM
011	STRIJK BV.	BESSENLAAN 1	LISSE
013	SPITMAN EN ZN.	ACHTERTUIN 9	AALSMEER
014	DEZAANER L.J.A.	DE GRONDEN 101	LISSE
019	MOOIWEER FA.	VERLENGDE ZOMERSTR. 24	AALSMEER
020	BLOEM L.Z.H.W.	LINNAEUSHOF 17	HILLEGOM
021	TRA.A.	KOELEPLEKSTRAAT 10	LISSE
022	ERICA BV.	BERKENWEG 87	HEEMSTEDE
034	DE GROENE KAS BV.	GLASWEG 1	AALSMEER
035	FLORIA BV.	OOVENSTRAAT 70	AALSMEER
453	Pieter		

INSERT INTO



```
1 • use bieren;
2
3 • insert into soorten (soortnr, soort)
4 values (30, 'Extra donker');
5
6 • insert into Brouwers
7 values (99, 'Brouwerij Vaattappers',
8 'Interleuvenlaan 2', 3000, 'Heverlee', 1000);
```

Opmerking

Als de primaire sleutel van een tabel een **Autonummerveld** (Auto increment) is, dan geven we dit veld niet op.

SELECT INTO

Syntaxis

```
SELECT column1, column2 INTO table_name (column1, column2, ...)  
  FROM source_table_name  
  WHERE conditions;
```

--maakt een kopie van de tabel bieren en kopieert alle rijen

```
SELECT * into BierenKopie from Bieren
```

--maakt een lege kopie van de tabel bieren

```
SELECT * into BierenKopie2 from Bieren where 1=2;
```

--voeg alle rijen van bieren toe aan bierenKopie2

INSERT INTO.... SELECT FROM

Syntaxis

```
INSERT INTO table_name (column1, colomn2, ...)  
  SELECT column1, colomn2  
  FROM source_table_name  
  WHERE conditions;
```

```
USE bieren;  
INSERT INTO soortenkopie (soortnr,soort)  
SELECT * FROM soorten;
```

of korter...

```
USE bieren;  
INSERT INTO soortenkopie SELECT * FROM soorten;
```

```
--voeg alle rijen van bieren toe aan bierenKopie2  
INSERT INTO BierenKopie2 select * from bieren;
```

UPDATE

Waarde(n) in 1 enkele kolom aanpassen:

Syntaxis

```
UPDATE table_name  
SET column_name = value  
WHERE condition
```

Waarde(n) in meerdere kolommen aanpassen:

Syntaxis

```
UPDATE table_name  
SET column_name1 = value1,  
    column_name2 = value2  
WHERE condition;
```

```
update brouwers set adres = 'Biervatlaan 12', PostCode=1000  
, gemeente = 'Brussel'  
where brouwernr = 99;
```

UPDATE waarbij waarden vanuit andere tabel komen

Syntaxis 1ste methode

```
UPDATE table_name1 INNER JOIN table_name2  
ON table_name1.column_name = table_name2.column_name  
SET column_name = expression  
WHERE condition;
```

Syntaxis 2de methode

```
UPDATE table_name  
SET column_name = expression  
WHERE ... IN (SELECT... );
```

UPDATE waarbij waarden vanuit andere tabel komen

```
UPDATE table_name1 INNER JOIN table_name2  
ON table_name1.column_name = table_name2.column_name  
SET column_name = expression  
WHERE condition;
```

```
UPDATE Bieren INNER JOIN Brouwers  
ON Bieren.BrouwerNr = Brouwers.BrouwerNr  
SET Alcohol = Alcohol + 0.5  
WHERE Brouwers.BrNaam = "Artois"
```

UPDATE waarbij waarden vanuit andere tabel komen

Syntaxis 2de manier

```
UPDATE table_name  
SET column_name = expression  
WHERE ... IN (SELECT ... );
```

```
UPDATE bierenkopie2  
SET alcohol = alcohol + 0.5  
WHERE BrouwerNr IN  
( SELECT BrouwerNr  
FROM Brouwers  
WHERE Brouwers.BrNaam = 'Artois' );
```

UPDATE waarbij waarden vanuit andere tabel komen – ook mogelijk met subqueries

Bijvoorbeeld:

```
update bierenkopie2
set alcohol = maxAlcoholPersoort.maxi
from (select b.SoortNr,
             max(b.alcohol) as maxi
       from bieren b
      group by b.soortNr) as maxAlcoholPersoort
where bierenkopie2.SoortNr =
maxAlcoholPersoort.SoortNr;
```

DELETE

Syntax

```
DELETE FROM table_name  
WHERE condition
```

LES3-Slides.sql - (...SF67LSQ\hcopp (68))*

```
delete from soorten where soortnr = 30;
```

133 %

Oefeningen

https://www.w3schools.com/sql/exercise.asp?filename=exercise_insert1

https://www.w3schools.com/sql/exercise.asp?filename=exercise_update1

https://www.w3schools.com/sql/exercise.asp?filename=exercise_update2

https://www.w3schools.com/sql/exercise.asp?filename=exercise_update3

https://www.w3schools.com/sql/exercise.asp?filename=exercise_delete1

https://www.w3schools.com/sql/exercise.asp?filename=exercise_delete2

oefeningen AdresTabel (Database AdresDb)

1. Voeg jouw naam aan de tabel toe;
2. Verwijder Lieve Thorens (met één R) uit de tabel, dit was een schrijffout;
3. Verander het adres van Mia Groter naar Stationstraat 25 en het telefoonnummer naar 016-85 21 25;
4. Voeg jullie instructeur toe aan de tabel, maar enkel de voornaam en familienaam. De rest is privé;

DATABANKEN – SQL - DDL

DDL: Tabellen en relaties

CREATE TABLE

--DDL

--CREATE TABLE

use BierenDb;

```
CREATE TABLE klanten(  
  KlantNr int IDENTITY(1,1) NOT NULL PRIMARY KEY,  
  KlNaam nvarchar(50) NOT NULL,  
  KlAdres nvarchar(100) NOT NULL,  
  KlPostCode nvarchar(4) NOT NULL,  
  KlEmail nvarchar(15) NULL);
```

--Nieuwe tabel aanmaken vanaf bestaande

--met data

```
SELECT KlantNr, KlNaam into TblKlantNamen  
FROM klanten;
```

DROP TABLE

Deze instructie verwijdt een bestaande tabel uit een database.

```
DROP TABLE tabel
```

De instructie DROP bevat de volgende onderdelen:

Onderdeel	Beschrijving
Tabel	De naam van de tabel die je wilt verwijderen

Voordat je een tabel kunt verwijderen, moet je de tabel sluiten.

```
DROP TABLE klanten
```

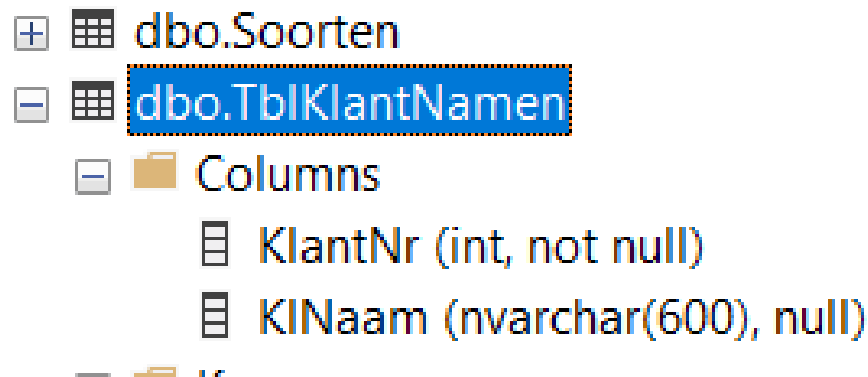
verwijdert de tabel klanten uit database.

ALTER TABLE

```
ALTER TABLE TblKlantNamen  
ALTER COLUMN KlNaam nvarchar(600);
```

```
ALTER TABLE TblKlantNamen  
ADD KlgebDatum Date NULL;
```

```
ALTER TABLE TblKlantNamen  
DROP COLUMN KlgebDatum;
```



CONSTRAINT

= een **beperkende voorwaarde** voor één of meerdere kolommen in de instructies **ALTER TABLE** en **CREATE TABLE**

CREATE TABLE + CONSTRAINT Syntax:

```
CREATE TABLE table_name
(
  column_name1 data_type(size) constraint_name,
  column_name2 data_type(size) constraint_name,
  column_name3 data_type(size) constraint_name,
  ....
);
```

CONSTRAINTS

- **NOT NULL** - Een kolom kan geen NULL waarde bevatten
- **UNIQUE** - Iedere rij heeft een uniek veld in deze kolom.
- **PRIMARY KEY** - Een combinatie van NOT NULL en UNIQUE.
- **FOREIGN KEY** - referentiedata naar een veld in een ander tabel.
- **CHECK** - De veldwaardes in deze kolom moeten aan bepaalde voorwaarden voldoen.
- **DEFAULT** - Specificeert de standaardwaarde wanneer een veld niet werd ingevuld.

CONSTRAINTS - Voorbeelden
























--CONSTRAINTS

```
CREATE TABLE Klanten (  
  KlantNr int IDENTITY(1,1) NOT NULL PRIMARY KEY,  
  RijkRegNr char(11) NOT NULL UNIQUE,  
  VoorNaam varchar(255) NOT NULL,  
  FamilieNaam varchar(255),  
  FavorietBierNr int FOREIGN KEY REFERENCES Bieren(BierNr),  
  Leeftijd smallint NULL,  
  CONSTRAINT CHK_LeeftijdKlant CHECK (Leeftijd>=18)  
);
```

```
ALTER TABLE Klanten  
DROP CONSTRAINT CHK_LeeftijdKlant;
```

```
ALTER TABLE Klanten  
ADD CONSTRAINT CHK_LeeftijdKlant CHECK (Leeftijd>=16);
```

CONSTRAINTS – Voorbeelden vervolg

- [-]  BierenDb
 - [+]  Database Diagrams
 - [-]  Tables
 - [+]  System Tables
 - [+]  FileTables
 - [+]  External Tables
 - [+]  Graph Tables
 - [+]  dbo.Bieren
 - [+]  dbo.Brouwers
 - [-]  dbo.Klanten
 - [-]  Columns
 -  KlantNr (PK, int, not null)
 -  RijkRegNr (char(11), not null)
 -  VoorNaam (varchar(255), not null)
 -  FamilieNaam (varchar(255), null)
 -  FavorietBierNr (FK, int, null)
 -  Leeftijd (smallint, null)
 - [-]  Keys
 -  PK_Klanten_A261F83D4EDA86B1
 -  FK_Klanten_Favorie_74AE54BC
 -  UQ_Klanten_B4B9C4373412341C
 - [-]  Constraints
 -  CHK_LeeftijdKlant

FOREIGN KEY CONSTRAINTS

De optionele clausule `ON DELETE` bepaalt wat er gebeurt als het record in de *parent table* verwijderd wordt, `ON UPDATE` bepaalt wat er gebeurt als de foreign key waarde in de *parent table* gewijzigd wordt. De mogelijke settings zijn:

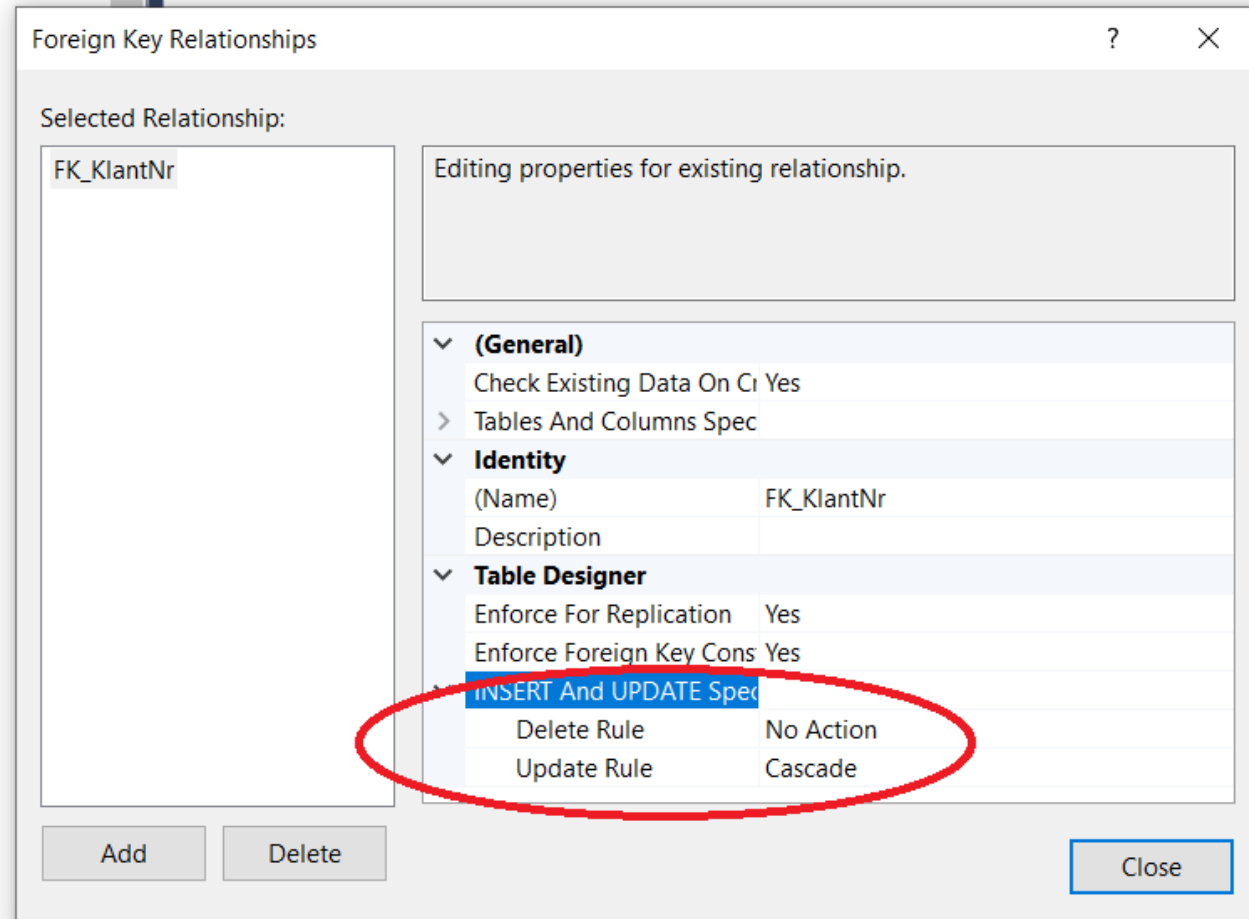
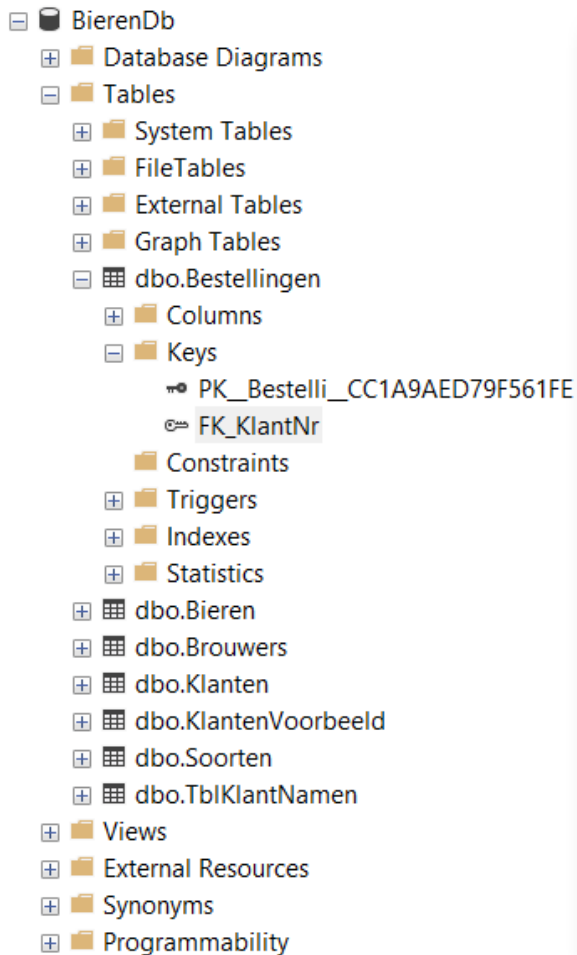
- **NO ACTION:** `DELETE` of `UPDATE` voor de *parent table* wordt geweigerd, er gebeurt niets.
Dit is ook de standaardinstelling als de clausules `ON DELETE/UPDATE` niet gespecificeerd zijn.
- **SET NULL:** bij een `DELETE` of `UPDATE` van de *parent table* wordt de waarde van de gerefereerde kolom(men) in de *child table* op `NULL` gezet. Deze kolommen mogen uiteraard geen `NOT NULL` ingesteld hebben
- **CASCADE:** bij een `DELETE` van een record in de *parent table* wordt dit record verwijderd en automatisch ook de gerelateerde rijen van de *child table*. Bij een update van een record in de *parent table* wordt dit record gewijzigd en automatisch ook de gerelateerde rijen van de *child table* bijgewerkt.

FOREIGN KEY CONSTRAINTS -Voorbeeld

De 1-N relatie tussen klant en bestellingen.aanmaak klanten:

```
-- FOREIGN KEY CONSTRAINT
CREATE TABLE Bestellingen
(
    BestelNr int IDENTITY(1,1) NOT NULL PRIMARY KEY,
    KlantNr int NOT NULL,
    BestelDatum datetime NOT NULL,
    CONSTRAINT FK_KlantNr FOREIGN KEY(KlantNr) REFERENCES Klanten(KlantNr)
    ON DELETE NO ACTION
    ON UPDATE CASCADE
);
```

FOREIGN KEY Constraint – vervolg voorbeeld



DROP FOREIGN KEY

De constraint heeft een naam **FK_klantNr**
Als we die willen wissen is het eenvoudig:

```
ALTER TABLE Bestellingen DROP CONSTRAINT FK_KlantNr;
```

FOREIGN KEY: nog een voorbeeld

De relatie later toevoegen:

```
DROP TABLE IF EXISTS Bestellingen;
```

```
CREATE TABLE Bestellingen  
(  
    BestelNr int IDENTITY(1,1) NOT NULL PRIMARY KEY,  
    KlantNr int NULL,  
    BestelDatum datetime NOT NULL  
);
```

```
ALTER TABLE Bestellingen  
ADD FOREIGN KEY (KlantNr) REFERENCES Klanten(KlantNr)  
ON UPDATE SET NULL;
```

CREATE INDEX

Indexen laten de database toe de data sneller te vinden zonder de gehele tabel te lezen.

SQL CREATE INDEX Syntax

(maakt een index aan. Dubbele waarden zijn toegestaan)

```
CREATE INDEX index_name  
ON table_name (column_name)
```

SQL CREATE UNIQUE INDEX Syntax

(maakt een unieke index aan. Geen dubbele waarden !)

```
CREATE UNIQUE INDEX index_name  
ON table_name (column_name)
```

CREATE INDEX - Voorbeelden

--CREATE INDEX

```
CREATE INDEX IDX_BestelDatum  
ON Bestellingen (BestelDatum);
```

```
CREATE UNIQUE INDEX IDX_RijkRegNr  
ON Klanten (RijkRegNr);
```

CREATE INDEX - Voorbeelden -Vervolg

[-] [Table Icon] dbo.Bestellingen

[+] [Folder Icon] Columns

[+] [Folder Icon] Keys

[Folder Icon] Constraints

[+] [Folder Icon] Triggers

[-] [Folder Icon] **Indexes**

[+] [Index Icon] **IDX_BestelDatum (Non-Unique, Non-Clustered)**

[+] [Key Icon] PK__Bestelli__CC1A9AED20E4AC9F (Clustered)

[-] [Table Icon] dbo.Klanten

[+] [Folder Icon] Columns

[+] [Folder Icon] Keys

[+] [Folder Icon] Constraints

[+] [Folder Icon] Triggers

[-] [Folder Icon] Indexes

[+] [Index Icon] **IDX_RijkRegNr (Unique, Non-Clustered)**

[+] [Key Icon] PK__Klanten__A261F83D4EDA86B1 (Clustered)

Oefeningen DDL

Download van Github:

Oefening-DDL-Bedrijf.pdf

Oefening ALTER Table.pdf

SQL-Oefeningen CreateTable.pdf

DATABANKEN - SQL

Views

VIEWS

= Een virtuele tabel die het resultaat is van een SQL-statement.

Een virtuele tabel is als een echte tabel met rijen, kolommen en velden.

De velden in een view kunnen uit één of meerdere echte tabellen komen

CREATE VIEW

Maakt een nieuwe view aan

SQL CREATE VIEW Syntax:

```
CREATE VIEW view_name AS  
SELECT column_name(s)  
FROM table_name  
WHERE condition
```

DROP VIEW

Verwijdert een view

SQL DROP VIEW Syntax:

DROP VIEW view_name

SQL – SAMENVATTING

- **Wijzigen van data**
 - **Insert - update - delete**
- **DDL**
- **Tabellen en relaties**
- **Views**

Oefeningen

Download van Github:

Demo-View.sql

Oefeningen Views_3.pdf

VRAGEN?