howest
hogeschool

Syllabus

Databases

Mileto Di Marco
Kristien Roels
Bart Soete

Inhoudsopgave

H7 - Het datamodel implementeren m.b.v. SQL

1 Inleiding

In de vorige hoofdstukken werden de SQL DML-statements (DML = Data Manipulation Language) SELECT, INSERT, UPDATE en DELETE behandeld voor het raadplegen, toevoegen, wijzigen en verwijderen van data.

In dit hoofdstuk worden de SQL DDL-statements (DDL = Data Definition Language) CREATE, ALTER en DROP behandeld voor:

- Het aanmaken (en wijzigen en verwijderen) van een databank
- Het aanmaken (en wijzigen en verwijderen) van tabellen
- Het aanmaken (en wijzigen en verwijderen) van indexen

De Microsoft Help-pagina's resp. voor maken van de databank, de tabellen en de indexen:
https://docs.microsoft.com/nl-be/sql/t-sql/statements/create-database-sql-server-transact-sql
https://docs.microsoft.com/nl-be/sql/t-sql/statements/create-table-transact-sql
https://docs.microsoft.com/nl-be/sql/t-sql/statements/create-index-transact-sql

2 Een databank maken, wijzigen en verwijderen

Een databank is de fysieke container voor het databankschema, de data en alle server-side programmering.

2.1 Een databank maken

Een databank maken gebeurt met het CREATE DATABASE-statement. De eenvoudigste vorm van dit statement heeft de volgende syntax:

CREATE DATABASE databank

Dit CREATE DATABASE-statement maakt een databank aan met de vermelde naam.

Maak een databank gradtest aan.
create database gradtest

Na een refresh van de Object Explorer kan de databanknode bekeken worden.

Rechtsklik op de databanknode, kies Properties, Files.
De databank bestaat uit 2 files: een datafile (gradtest.mdf) en een transaction log (gradtest_log.ldf).

De datafile bevat alle data: systeemtabellen, gebruikerstabellen, indexen, views, stored procedures, user-defined functions, triggers en security-rechten.

In de transaction log wordt elke INSERT, UPDATE of DELETE weggeschreven vóóraleer de handelingen in de datafile gebeuren. Men spreekt over een write-ahead transaction log. Deze write-ahead transaction log wordt gebruikt om de databank te kunnen herstellen na een failure.

Bekijk de default instellingen voor zowel de datafile (gradtest.mdf) als de transaction log (gradtest_log.ldf):

Path: C:\Program Files\Microsoft SQL Server\MSSQL14.SQLEXPRESS\MSSQL\DATA
Initial Size (MB): 8
Autogrowth / Maxsize: By 64 MB, Unlimited

Het is ook mogelijk om het CREATE DATABASE-statement te voorzien van een aantal instellingen. De syntax:

CREATE DATABASE databank
ON-clausule
LOG ON-clausule

De ON-clausule specifieert waar de datafile moet opgeslagen worden.

De LOG ON-clausule specifieert waar de transaction log moet opgeslagen worden.

Beide clausules hebben de volgende instellingsmogelijkheden:
Name: bevat de logische filename.
Filename: bevat de fysische filename.
Size: specifieert de initiële size van de file.
Maxsize: specifieert de size tot waar de file kan groeien. Indien niet gespecifieerd, kan de file de volledige schijf volschrijven.
Filegrowth: specifieert de hoeveelheid ruimte waarmee de file groeit. Indien niet gespecifieerd, gebeurt de groei per 10%.

Bekijk het onderstaande CREATE DATABASE-statement met een ON- en LOG ON-clausule.

```
create database gradtest2
on
( name = 'gradtest2',
filename = 'C:\SQLData\gradtest2.mdf',
size = 10MB,
maxsize = 200GB,
filegrowth = 100MB )
log on
( name = 'gradtest2_log',
filename = 'D:\SQLLog\gradtest2_log.ldf',
size = 5MB,
maxsize = 10GB,
filegrowth = 100MB )
```

Zoals reeds besproken in hoofdstuk 3 kan een databank ook gemaakt worden in de SQL Server Management Studio. Rechtsklik daartoe op de Databases-node en kies in het snelmenu New Database …. 3 tabbladen kunnen ingevuld worden: General, Options en Filegroups.

Merk ook op dat de gegenereerde code kan bekeken worden. Rechtsklik daartoe op de databanknode en kies in het snelmenu Script Database as, CREATE To, New Query Editor Window.

2.2 Een databank wijzigen

Een databank wijzigen gebeurt met het ALTER DATABASE-statement. Er kunnen heel wat instellingen gewijzigd worden (zie de F1-Help: https://docs.microsoft.com/nl-nl/sql/t-sql/statements/alter-database-transact-sql?view=sql-server-ver15). In deze cursus beperken we ons tot het wijzigen van de databanknaam. De syntax is dan:

ALTER DATABASE databank
MODIFY NAME = nieuwenaam

De vermelde databank krijgt een nieuwe naam.

Wijzig de naam van de databank gradtest naar graddatabank.
alter database gradtest modify name = graddatabank

2.3 Een databank verwijderen

Een databank verwijderen gebeurt met het DROP DATABASE-statement. Merk op dat de systeemdatabanken (msdb, master, model, tempdb) niet verwijderd kunnen worden. De syntax:

DROP DATABASE [IF EXISTS] databank

De vermelde databank wordt verwijderd: alle diskfiles die gebruikt worden door de databank worden verwijderd. Als de vermelde databank niet bestaat, wordt een foutmelding gegenereerd, tenzij IF EXISTS vermeld wordt.

IF EXISTS gaat eerst na of de databank bestaat vooraleer deze te verwijderen.

Verwijder de databank graddatabank.
drop database if exists graddatabank

Let op: als de databank in gebruik is, kan ze niet onmiddellijk verwijderd worden met drop database. Test dit uit.
--Maak eerst de databank opnieuw aan: create database graddatabank --Neem ze in gebruik: use graddatabank --Probeer ze te verwijderen: drop database if exists graddatabank →Foutmelding: Cannot drop database "graddatabank" because it is currently in use.

Verwijder een databank die in gebruik is.
--Maak van de systeemdatabank master de actieve databank: use master --Zorg dat je exclusieve toegang hebt tot de databank die je wil verwijderen: alter database graddatabank set single_user with rollback immediate --Nu kan de databank verwijderd worden: drop database if exists graddatabank --Verifieer, na een refresh, de Object Explorer.

3 Tabellen maken, wijzigen en verwijderen

In dit deel worden tabellen gemaakt, gewijzigd en verwijderd in de databank graddatabank. Maak dus eerst deze databank (CREATE DATABASE graddatabank) en zorg dat deze databank de actieve databank is (USE graddatabank).

3.1 Tabellen maken

Een tabel maken gebeurt met het CREATE TABLE-statement. De globale syntax van het CREATE TABLE-statement ziet er als volgt uit:

CREATE TABLE tabel (
kolom datatype [constraints, default, properties, ...]
[, …]
[, constraints]
)

> *Dit CREATE TABLE-statement maakt een tabel aan met de vermelde naam. De kolomdefinities bepalen de structuur van de tabel.*
>
> *Het datatype van een kolom moet zorgvuldig gekozen worden. Het is de eerste vorm om data integriteit af te dwingen. In hoofdstuk 3 staat een overzicht van de mogelijke datatypes per categorie.*

Maak een tabel Studenten (studentnummer, familienaam, voornaam, geboortedatum, inschrijvingsgeld)
create table Studenten (studentnummer uniqueidentifier, familienaam nvarchar(50), voornaam nvarchar(50), geboortedatum date, inschrijvingsgeld smallint)
insert into Studenten(studentnummer, familienaam, voornaam, geboortedatum, inschrijvingsgeld) values (newid(), 'Delange', 'Peter', '1985-06-20', 500)

Een DEFAULT waarde kan opgegeven worden bij een kolom. Deze waarde zal gebruikt worden als er geen waarde gespecifieerd wordt voor deze kolom in het INSERT-statement.

Tabel Studenten: geef aan de kolommen studentnummer en inschrijvingsgeld een default waarde.
drop table if exists Studenten
create table Studenten (studentnummer uniqueidentifier default newid(), familienaam nvarchar(50), voornaam nvarchar(50), geboortedatum date, inschrijvingsgeld smallint default 400)
insert into Studenten(familienaam, voornaam, geboortedatum) values('Delange', 'Peter', '1985-06-20')

Constraints zijn voorwaarden waaraan de data moet voldoen. Constraints dwingen data integriteit af; d.w.z. ze zorgen ervoor dat de data die in de tabel komt, correct is (aan de voorwaarden voldoet).

Kolomconstraints staan bij een kolomdefinitie en refereren naar die kolom.

Tabelconstraints staan na de kolomdefinities en kunnen naar meerdere kolommen uit de tabel refereren.

Bij elke INSERT, UPDATE of DELETE worden de constraints geverifieerd. Indien aan de voorwaarden voldaan is, dan wordt de INSERT, UPDATE of DELETE uitgevoerd. In het andere geval komt er een foutmelding en gebeurt de INSERT, UPDATE of DELETE niet!

Het verhoogt de leesbaarheid als aan de constraints een zinvolle naam wordt gegeven (SQL Server geeft zelf wel vrij duidelijke namen).

Er zijn 5 types constraints:

- NOT NULL constraint : verzekert dat de kolom geen NULL-waarden bevat.
- PRIMARY KEY constraint : verzekert een unieke niet-NULL-sleutel.
- FOREIGN KEY constraint : verzekert dat de waarde voorkomt in de gerelateerde primary key.
- UNIQUE constraint : verzekert een unieke waarde
- CHECK constraint : verzekert dat aan de voorwaarde voldaan is.

De NULL / NOT NULL constraint geeft aan of de kolom de NULL-waarde aanvaardt of niet.

Tabel Studenten: studentnummer, familienaam, voornaam en inschrijvingsgeld móeten ingevuld worden.
drop table if exists Studenten
create table Studenten (studentnummer uniqueidentifier default newid() not null, familienaam nvarchar(50) not null, voornaam nvarchar(50) not null, geboortedatum date, inschrijvingsgeld smallint default 400 not null)

De PRIMARY KEY constraint definieert een primaire sleutel. Een primaire sleutel is een kolom (kolommen) die een rij uniek identificeert (identificeren) binnen de tabel. De primary key constraint dwingt entiteit integriteit af. Er kan slechts 1 primary key constraint per tabel zijn. Alle kolommen in een primary key moeten NOT NULL gedefinieerd zijn.

Twee datatypes zijn uitstekend voor primaire sleutels:

- Een uniqueidentifier kolom bevat een GUID (Globally Unique Identifier). Een GUID is een 16 Byte hexadecimaal getal dat uniek is over de hele wereld. De GUID-generator maakt gebruik van de NIC-code van de computer, het MAC-adres, de current tick van de klok, … om de uniciteit te bekomen.
- Kolommen met de property Identity. SQL Server genereert een waarde voor de kolom als een rij ingevoegd wordt. De Identity property kan geplaatst worden bij tinyint, smallint, int, bigint, decimal(p,0) of numeric(p,0) kolommen. Er kan slechts 1 Identity kolom per tabel gemaakt worden. Een default-constraint kan niet samen met een Identity-kolom gebruikt worden.

Tabel Studenten: definieer een primaire sleutel. Gebruik een Identity-kolom.
drop table if exists Studenten
--Met kolomconstraint create table Studenten (studentnummer int identity not null primary key, familienaam nvarchar(50) not null, voornaam nvarchar(50) not null, geboortedatum date, inschrijvingsgeld smallint default 400 not null)
--Of met tabelconstraint create table Studenten (studentnummer int identity not null, familienaam nvarchar(50) not null, voornaam nvarchar(50) not null, geboortedatum date, inschrijvingsgeld smallint default 400 not null, constraint pk_Studenten primary key(studentnummer))
Als de PK enkelvoudig is (1 kolom), dan kies je hoe je de PK definieert: direct bij de kolom als kolomconstraint, of onderaan als tabelconstraint. Als de PK echter samengesteld is (meerdere kolommen), dan moet je de PK als tabelconstraint definiëren (je kan niet naast meerdere kolommen primary key zetten).

De FOREIGN KEY constraint definieert een vreemde sleutel. Een vreemde sleutel is een kolom (kolommen) waarvan de waarde moet voorkomen in de lijst van waarden in de gerelateerde primaire sleutel of unieke kolom (of NULL is). Een vreemde sleutel verbindt 2 tabellen. De Foreign key constraint dwingt referentiële integriteit af (de referentie heeft integriteit, is correct).

Tabel Studenten en tabel Departementen. Een student volgt les in één departement.
drop table if exists Studenten
create table Departementen (deptnummer smallint identity not null primary key, deptnaam nvarchar(50) not null, stad nvarchar(50))
--Met kolomconstraint create table Studenten (studentnummer int identity not null primary key, familienaam nvarchar(50) not null, voornaam nvarchar(50) not null, geboortedatum date, inschrijvingsgeld smallint default 400 not null, deptnummer smallint not null foreign key references Departementen(deptnummer))
--Of met tabelconstraint

```
create table Studenten
(
studentnummer int identity not null,
familienaam nvarchar(50) not null,
voornaam nvarchar(50) not null,
geboortedatum date,
inschrijvingsgeld smallint default 400 not null,
deptnummer smallint not null,
constraint pk_Studenten primary key(studentnummer),
constraint fk_Studenten foreign key(deptnummer) references Departementen(deptnummer)
)
```

Bij de Foreign Key constraint kan de clausule ON DELETE CASCADE geplaatst worden. Als dan een rij uit de parent tabel verwijderd wordt, zullen de overeenkomstige rijen uit de refererende tabel ook verwijderd worden.

Analoog kan ook de clausule ON UPDATE CASCADE toegevoegd worden.

Tabel Studenten en tabel Departementen: wat is het effect van ON DELETE CASCADE bij de vreemde sleutel?

Bij het verwijderen van een departement, worden al de studenten uit dat departement verwijderd.

Primary Key (PK) en Foreign Key (FK) zijn zéér belangrijke begrippen bij een relationele databank. In een relationele databank:

- definieer in elke tabel een PK
- definieer de nodige FKs: deze zorgen dat de gegevens uit de verschillende tabellen op een correcte manier gelinkt zijn aan elkaar.

De UNIQUE constraint verzekert een unieke waarde in een niet-primaire sleutelkolom. Null-waarden zijn niet toegelaten in de kolom.

Tabel Departementen: eis dat de naam van een departement uniek is.

```
drop table if exists Studenten
drop table if exists Departementen
```

```
create table Departementen
(
deptnummer smallint identity not null primary key,
deptnaam nvarchar(50) not null unique,
stad nvarchar(50)
)
```

De CHECK constraint zorgt dat aan de opgegeven voorwaarde voldaan is. De constraint kan geen data uit andere tabellen benaderen.

Tabel Studenten: de voornaam mag niet beginnen met een S en het geboortejaar moet > 1980 zijn.

```
drop table if exists Studenten
```

```
create table Studenten
(
studentnummer int identity not null,
```

```
familienaam nvarchar(50) not null,
voornaam nvarchar(50) not null check(voornaam not like 'S%'),
geboortedatum date check(year(geboortedatum) > 1980),
inschrijvingsgeld smallint default 400 not null,
deptnummer smallint not null,
constraint pk_Studenten primary key(studentnummer),
constraint fk_Studenten foreign key(deptnummer) references Departementen(deptnummer)
)
```

3.2 Tabellen wijzigen

Het wijzigen van de tabelstructuur gebeurt met het ALTER TABLE-statement. Dit is een zeer uitgebreid statement (zie de F1-Help https://docs.microsoft.com/en-us/sql/t-sql/statements/alter-table-transact-sql?view=sql-server-ver15): kolommen en constraints kunnen gewijzigd, toegevoegd en verwijderd worden. In deze cursus beperken we ons tot de meest voorkomende wijzigingen. De globale syntax van het ALTER TABLE-statement ziet er als volgt uit:

```
ALTER TABLE tabel
    ALTER COLUMN kolom datatype [NULL | NOT NULL ]
|   ADD { kolomdefinitie | tabelconstraint } [ , … ]
|   DROP { COLUMN [ IF EXISTS ] kolom | CONSTRAINT [ IF EXISTS ] constraint } [ , … ]
```

Dit ALTER TABLE-statement wijzigt de tabelstructuur van de tabel met de vermelde naam.

In de ALTER COLUMN-clausule wordt het datatype en eventueel de NOT NULL-constraint gewijzigd.

In de ADD-clausule wordt een kolom of een tabelconstraint toegevoegd. Vaak worden tabelconstraints op deze manier toegevoegd (nadat de tabel gemaakt werd met het CREATE TABLE-statement).

In de DROP-clausule wordt een kolom of een constraint verwijderd.

Tabel Studenten: voeg een constraint toe op de kolom inschrijvingsgeld.

```
alter table Studenten
add constraint ch_inschrijvingsgeld CHECK(inschrijvingsgeld between 0 and 500)
```

Tabel Studenten: voeg een kolom geslacht toe. Waarom is default hier toegevoegd?

```
alter table Studenten
add geslacht char(1) NOT NULL DEFAULT 'M'
```

DEFAULT is toegevoegd omdat bij de reeds eerder ingegeven studenten het geslacht niet ingevuld zou zijn (en dat is in tegenspraak met de NOT NULL-constraint).

Tabel Studenten: pas de kolom geslacht aan: de lengte wordt 2 karakters.

```
alter table Studenten
alter column geslacht char(2)
```

Tabel Studenten: verwijder de check-constraint bij inschrijvingsgeld.

```
alter table Studenten
```

drop constraint ch_inschrijvingsgeld

Tabel Studenten: verwijder de kolom geslacht.
alter table Studenten drop column geslacht

3.3 Tabellen verwijderen

Het verwijderen van een tabel gebeurt met het DROP TABLE-statement. De syntax:

DROP TABLE [IF EXISTS] tabel

> *De vermelde tabel wordt* verwijderd. *Als de vermelde tabel niet bestaat, wordt een foutmelding gegenereerd, tenzij IF EXISTS vermeld wordt. Als er naar de tabel verwezen wordt vanuit een vreemde sleutel, dan kan de tabel niet verwijderd worden (eerst moet de Foreign Key constraint verwijderd worden).*
>
> *IF EXISTS gaat eerst na of de tabel bestaat vooraleer deze te verwijderen.*

Merk op dat de systeemtabellen niet verwijderd kunnen worden.

Verwijder de tabel Studenten.
drop table if exists Studenten

4 Indexen maken, wijzigen en verwijderen

In hoofdstuk 3 werd reeds het principe van indexen besproken.

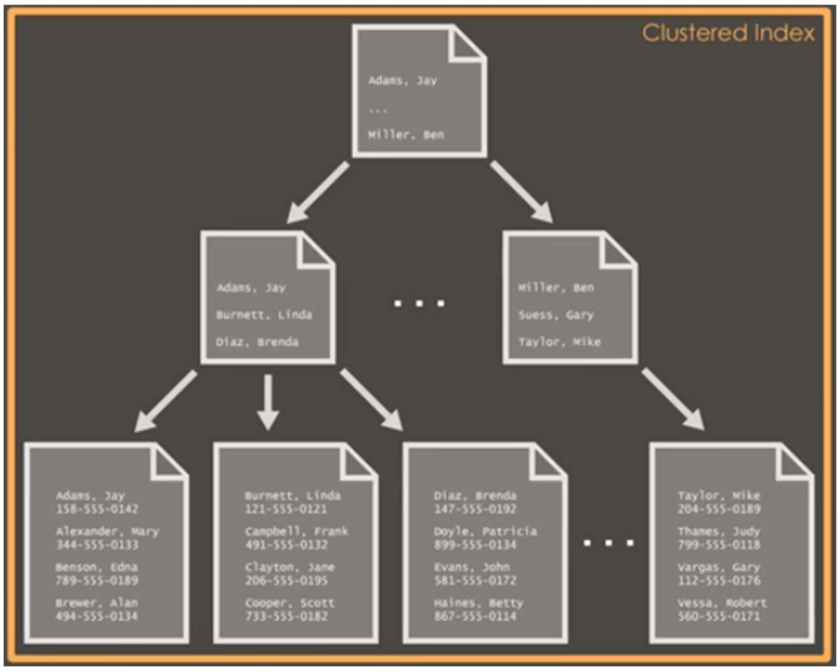
Even opfrissen: een index is een bijkomende datastructuur die het opzoeken van data in de tabel (drastisch) versnelt. Het toevoegen en verwijderen van data zal echter trager verlopen omdat de index moet aangepast worden. Anders gezegd: indexen verhogen de search performantie (vaak spectaculair) en verlagen de update performantie.

Het is interessant om een index te maken op een kolom waarop er vaak gezocht wordt én het is ook interessant om een index te maken op de foreign key.

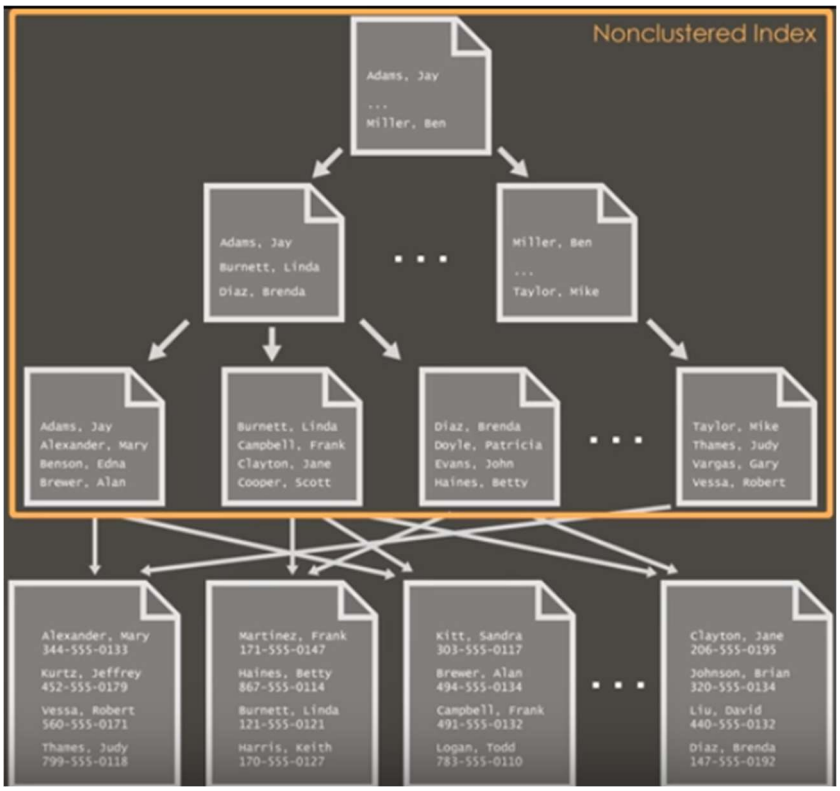
Het gebruik en onderhoud van een index wordt door SQL Server automatisch gedaan.

SQL Server kent 2 soorten indexen: clustered en non-clustered.

- Clustered index.
  Een clustered index determineert de fysieke volgorde van de datarijen in een tabel. Anders gezegd: als een clustered index gemaakt wordt op een kolom van een tabel, dan zal SQL Server de rijen van de tabel sorteren op die kolom.
  Er kan dus maar 1 clustered index zijn per tabel.
  Het onderste niveau van de clustered index (leaf) bevat de actuele datarijen.
  Een clustered index is als een telefoonboek.
  Grafische voorstelling van een clustered index op familienaam en voornaam:
  (https://www.youtube.com/watch?v=ITcOiLSfVJQ)



- Non-clustered index.
  Een non-clustered index wordt gescheiden van de data opgeslagen, is gesorteerd op de indexkolomm(men) en heeft pointers naar de eigenlijke data.
  Een non-clustered index beïnvloedt de volgorde van de datarijen niet.
  Er kunnen dus meerdere non-clustered indexen zijn per tabel.
  Een non-clustered index is als de index achteraan een boek.
  Grafische voorstelling van een non-clustered index op familienaam en voornaam:



4.1 Indexen maken

Het maken van een index gebeurt met het CREATE INDEX-statement. De syntax:

CREATE [UNIQUE] [CLUSTERED | NONCLUSTERED] INDEX index
ON tabel (kolom [ASC | DESC] [,...])

Dit CREATE INDEX-statement maakt een index op de kolom(men) van een tabel.

De optie UNIQUE maakt een unieke index. Een unieke index laat niet toe dat er 2 rijen zijn met dezelfde waarde in de indexkolom(men).

De optie CLUSTERED maakt een clustered index.

De optie NONCLUSTERED maakt een non-clustered index.

Merk op dat bij het definiëren van een primaire sleutel automatisch een unieke clustered index op de primaire sleutel gedefinieerd wordt. En bij het definiëren van een UNIQUE-constraint wordt automatisch een unieke index op deze kolom gedefinieerd.

select * from Studenten De resultaatset bevat de studenten volgens stijgend studentnummer. Hoe komt dit? Er staat geen ORDER BY-clausule.
Omdat er automatisch een clustered index werd gemaakt op de primaire sleutel studentnummer. En een clustered index bepaalt de fysieke volgorde van de rijen.

Tabel Studenten: maak een non-clustered index op familienaam.
create nonclustered index iFamilienaam on Studenten(familienaam)

4.2 Indexen wijzigen

Het wijzigen van een index kan met het ALTER INDEX-statement. Dit wordt niet behandeld in deze cursus. Voor een bespreking, zie de link hierboven naar de Help.

4.3 Indexen verwijderen

Het verwijderen van een index gebeurt met het DROP INDEX-statement. De syntax:

DROP INDEX [IF EXISTS] tabel.index

De vermelde index wordt verwijderd. *Als de vermelde index niet bestaat, wordt een foutmelding gegenereerd, tenzij IF EXISTS vermeld wordt. Merk op dat voor de indexnaam de tabelnaam moet vermeld worden.*

IF EXISTS gaat eerst na of de index bestaat vooraleer deze te verwijderen.

Tabel Studenten: verwijder de index iFamilienaam
drop index if exists Studenten.iFamilienaam