# DATABASES RELATIONEEL

| | | | |
|---|---|---|---|
| Week 1 | Les 1 | Normaliseren – Nulde + Eerste Normaalvorm | opdracht 1 |
| Week 2 | Les 1 | Normaliseren – Tweede + Derde Normaalvorm | opdracht 1 |
| Week 3 | Les 2 | ERD – Maken van een ERD | opdracht 2 |
| Week 4 | Les 3 | SQL – JOIN | opdracht 3.1 |
| Week 5 | Les 3 | SQL – Nested Queries/Sub-queries | opdracht 3.2 |
| Week 6 | Les 3 | SQL – Verdieping | opdracht 3.3 |

# LES 1: NORMALISEREN

Voor het aanmaken van een nieuwe database is het belangrijk dat we eerst goed nadenken over de structuur van een database. Er is veel theorie over te vinden hoe je op een juiste manier een database kan structureren.

Wat veel gebruikt wordt om een database visueel inzichtelijk te maken voordat we een daadwerkelijke database in elkaar geknutseld wordt is een ERD (Entity Relation Diagram). Voordat je het ERD gaat omzetten in werkelijke databasetabellen (in PHPMyAdmin), ga je eerst controleren of er geen fouten zitten in jouw database.

Het controleren en verbeteren van de database noemen we **normaliseren**.
Wat normaliseren is, wordt uitgelegd op Wikipedia:

- https://nl.wikipedia.org/wiki/Databasenormalisatie

Een belangrijk doel van normaliseren is het voorkomen van **redundantie**:

- https://infvo.github.io/db-0/html/glossary.html -> 'redundantie'

Redundantie is het bewaren van dezelfde gegevens op meerdere plaatsen (in meerdere tabellen). Soms is dat handig, maar meestal is het overbodig.

Bij het maken van een database moet je dus o.a. voorkomen dat gegevens dubbel bewaard worden. Daarom ga je normaliseren.

Normaliseren gebeurt in stappen:

- Nulde Normaalvorm:
  Dit is de database zoals deze er uit ziet voordat er gecontroleerd is; dus zonder normalisatie. In de meeste gevallen is dit enkel nog een tekstuele "casus".
- Eerste normaalvorm:
  De tabel voldoet aan de volgende eisen:
  - Alle velden in de tabel moeten enkelvoudig zijn. (Dus niet 2 gegevens in 1 veld)
  - Alle elementen in een kolom moeten hetzelfde soort gegeven voorstellen. (Dus in een datumveld staat geen telefoonnummer)
  - De kolommen binnen een tabel moeten unieke namen hebben. (In 1 tabel mag je niet 2 velden hebben met dezelfde naam)
  - Er mogen geen rijen voorkomen met volledig identieke gegevens. (Dus niet 2 precies dezelfde records).
- Tweede normaalvorm:
  Alle attributen, die niet in de sleutel opgenomen zijn, zijn afhankelijk van de volledige sleutel.
  Dus: als je de sleutel (key) van een record weet, dan weet je ook meteen alle

bijbehorende gegevens. Bijvoorbeeld: als je het studentnummer van iemand weet, dan weet je ook meteen de naam van de student, adres, klas, …

- Derde normaalvorm:
  De tabel staat in de tweede normaalvorm en bevat geen transitieve relaties.
  Voorbeeld: Student met nummer 87955 zit in klas d2b en heeft als mentor GTS.
  Het studentnummer bepaalt in welke klas de student zit, maar de klas bepaalt de mentor:
  studentnummer -> klas -> mentor
  Omdat het studentnummer niet de mentor bepaalt, mag mentor niet in de tabel STUDENT staan.

Er zij nog meer stappen voor het normaliseren van de database, maar die worden hier niet besproken. Normaal gesproken is een database na de derde normaalvorm in orde en kan deze goed gebruikt worden.

## OPDRACHT 1

Jullie gaan een database maken voor de schoolbibliotheek. De casus (het verhaal met de eisen en wensen van de bibliotheek) staat onder de opdracht.

Zet alle gegevens die bewaard moeten worden achter elkaar in een rij (of twee rijen) en bepaal de sleutel van de rij.
Daarna ga je normaliseren. Dit houdt meestal in dat je de rij(-en) die je hebt gaat splitsen in meerdere rijen, die later de tabellen gaan vormen.
Beschrijf voor elke stap (Nulde t/m Derde normaalvorm) wat je hebt veranderd aan de databasetabellen. Als je tijdens een normaalvorm niks hebt veranderd dan geef je dat ook aan.

Extra uitleg kan je vinden in:

- Het Word document “BC_Normaliseren” op Digitale Lesstof
- De PDF “Normaliseren” op Digitale Lesstof
- Online, bijvoorbeeld:
  - https://nl.wikipedia.org/wiki/Databasenormalisatie
  - https://www.jeroenmoonen.nl/blog/how-to/database-normaliseren-basisregels/

**DE CASUS:**

De schoolbibliotheek van het GLR leent boeken heeft een grote verzameling boeken met diverse thema's. Sommige boeken zijn geschikt voor de afdeling Media Vormgeven, andere boeken passen beter bij Softwaredevelopment. Alle boeken hebben daarom, naast een titel, schrijver, taal en ISBN, ook meerdere labels waarin staat voor welke afdelingen dit boek vooral geschikt is.

Studenten kunnen naar de bibliotheek om een boek te lenen. Een boek mag maximaal 2 weken in het bezit zijn van een student. Als deze langer wordt gehouden, dan krijgt de student een waarschuwingsmail. Na drie weken moet er een boete betaald worden; deze is 1 euro per week.

Op dit moment worden alle gegevens nog bewaard in een ouderwetse kaartenbak. De bibliotheek wil alles digitaliseren.
Alle gegevens van de boeken moeten in een database gezet worden. Het moet duidelijk zijn voor welke afdelingen een boek geschikt is.
Een boek moet online gereserveerd kunnen worden.
Als een student een boek komt lenen, wordt vastgelegd welk boek de student meeneemt en op welke datum dit gebeurt.
Van een student moet naast klas, studentnummer en naam, ook het emailadres bewaard worden; de student moet namelijk gewaarschuwd kunnen worden als hij/zij een boek te lang in het bezit heeft.
Aan de hand van de klas kan het systeem bepalen in welke afdeling de student zit; zo kan er makkelijk gezocht worden op geschikte boeken.

➔ Maak (op papier of in Word) alle tabellen die gemaakt moeten worden en bepaal per tabel de sleutel. Zorg dat de tabellen zijn genormaliseerd tot en met de derde normaalvorm.

## LES 2: OMZETTEN NAAR ERD

In de module van PROCES1 hebben jullie al geleerd hoe je een casus om moet zetten naar een ERD. Een 'casus' is een beschrijving van een bedrijf; het bedrijf wil de gegevens die beschreven staan in de casus bewaren in een database. Meestal zijn dat zoveel gegevens dat daar meerdere tabellen voor nodig zijn; deze tabellen hebben natuurlijk met elkaar te maken: ze hebben een relatie met elkaar.

- Voordat je de database gaat maken, moet je eerst uit de casus de entiteiten ('tabellen') en hun attributen ('velden') halen.
  Bepaal ook per entiteit de identifier (de 'primary key').
- Daarna bepaal je de relaties tussen de entiteiten. (1 - 1, 1 - ∞, ∞ - ∞)
- Afhankelijk van de relatie tussen 2 tabellen ga je de identifier (key) van de ene tabel in de andere tabel plaatsen. Op deze manier maak je een 'fysieke relatie' tussen de twee tabellen.

- Controleer aan het eind of alle entiteiten, attributen. Relaties en identifiers kloppen.
- Maak dan de volledige ERD in een ontwerpprogramma (als je dat nog niet gedaan hebt).

## OPDRACHT 2

In de vorige opdracht heb je de tabellen en de bijbehorende velden en sleutels bepaald.
Zet nu de tabellen in een ERD. Maak daarvoor gebruik van een (online) tool om alle entiteiten, attributen en relaties netjes in een schema te krijgen. Bijvoorbeeld Lucidchart.

**Extra uitleg** kan je vinden in:

- De reader "PROCES2 – ERD".
- Het Word document "BC_ERD" op Digitale Lesstof
- Online, bijvoorbeeld:
  - https://www.lucidchart.com/pages/nl/tutorial-database-structuur-en-ontwerp
  - https://creately.com/blog/nl/diagrammen/er-diagram-tutorial/

Doorloop ALLE STAPPEN die nodig zijn om een goed ERD te maken (zie reader "PROCES2 – ERD")
Leg alle stappen, inclusief het uiteindelijke ERD, vast in een Word document.

LET OP: Bepaal bij elke entiteit ('tabel') de juiste attributen ('velden') Het kan zijn dat je extra attributen moet toevoegen.
EN: geeft van elk attribuut aan wat voor datatype het is (VARCHAR, DATE, INT, …) en welk attribuut de primary key is.

## LES 3: SQL: STUCTURED QUERY LANGUAGE

Nu de database ontworpen, genormaliseerd en gerealiseerd is, kunnen we deze gaan gebruiken om gegevens toe te voegen, aan te passen, te verwijderen of uit te lezen.
We gaan ons nu vooral richten op de laatste: Hoe haal je de juiste informatie uit de database? Door de juiste vraag (**query**) te stellen, krijg je precies de gegevens die je nodig hebt en kan je daarmee direct aan het werk.

Om een vraag te stellen aan de datebase gebruiken we taal **SQL** (Structured Query Language). Dit is een uitgebreide taal met haar eigen syntax.

Op internet zijn diverse tutorials te vinden waarin uitgelegd wordt hoe SQL werkt.
Hieronder vind je een aantal:

- https://www.w3schools.com/sql/
- https://www.tutorialspoint.com/sql/index.htm
- https://gkoetsier.nl/tut_mysql.html **(Nederlands)**

- https://sqltutorial.nl/ (Nederlands)

In de opdracht wordt verwezen naar de reader over SQL: “BC_SQL.docx”.

## OPDRACHT 3.1

Tijdens de vorige opdrachten heeft iedereen een eigen database gemaakt met eigen gegevens. Tijdens deze opdracht begint iedereen met dezelfde, nieuwe database: “de tennisvereniging”.

Alle opdrachten worden uitgevoerd in het SQL-venster op PHPMyAdmin.

➔ Maak elke gevraagde query in het SQL-venster op PHPMyAdmin en test deze.
Daarna kopieer je de query naar jouw word-document!

### OPDRACHT 3a

1. Lees uit “BC_SQL.docx” de inleiding op bladzijde 1.
2. Importeer de SQL-code van de tennisvereniging (“tennis.sql”) in jouw eigen database op PHPMyAdmin.

### OPDRACHT 3b

1. Lees uit “BC_SQL.docx” de inleiding op bladzijde 2 t/m 5
   (“De FROM component”)
2. Maak opdracht 1 (blz. 5)
   Zet alle gemaakte queries in jouw word-document onder “opdracht3b”.

### OPDRACHT 3c

1. Lees uit “BC_SQL.docx” de inleiding op bladzijde 6 t/m 8
   (“JOINS”)
2. Maak opdracht 2 (blz. 9)
   LET OP: dit gaat over andere tabellen (zie reader)!
   Zet alle gemaakte queries in jouw word-document onder “opdracht3c”.

### OPDRACHT 3d

1. Lees uit “BC_SQL.docx” de inleiding op bladzijde 10 t/m 18
   (“De WHERE component”)
2. Maak opdracht 3 (blz. 18)
   Zet alle gemaakte queries in jouw word-document onder “opdracht3d”.

### OPDRACHT 3e

1. Lees uit “BC_SQL.docx” de inleiding op bladzijde 19 t/m 22
   (“De SELECT component”)
2. Maak opdracht 4 (blz. 22)
   Zet alle gemaakte queries in jouw word-document onder “opdracht3e”.