

# Normaliseren

## Leerdoel

In deze module leren we normaliseren. Normaliseren is het opzetten van een goed database design.

## ERD

In de praktijk vraag je wat een klant wil. De klant vertelt wat hij wil en jij zet dat om in een ERD.

In Database design level 1 heb je al kennis gemaakt met een ERD. Om je kennis op te halen kan je nog een keer naar de uitleg in e video kijken: <https://web.microsoftstream.com/video/c4c62660-1c4f-4c89-b09b-948cf7559c86>

## Waarom normaliseren?

Als je goed normaliseert en dus een goed datamodel maakt dan zorg je ervoor dat je gegevens zo worden opgeslagen dat:

- gegevens niet in tegenspraak zijn met elkaar
- gegevens maar één keer worden opgeslagen
- gegevens, maar op een plaats kunnen wijzigen

Wat je bijvoorbeeld niet wilt, is dat als je in een database iemand zijn geboortedatum vastlegt en dat je daarnaast ook de leeftijd vastlegt. Over een jaar of meer dan is iemand een jaar ouder en zijn de gegevens met elkaar in tegenspraak.

Of stel je hebt de prijs met en zonder BTW opgeslagen. Stel het BTW-tarief wijzigt dan moet je alle prijzen in de database opnieuw berekenen. Een goed database model zorgt ervoor dat dat niet hoeft.

Of stel je hebt iemand zijn naam verkeerd gespeld. Als je die aanpast in de database dan wil je dat één keer op één plek doen. Alleen op die manier weet je zeker dat je niets bent vergeten.

Zorgen dat gegevens maar één keer worden opgeslagen heet ook wel het voorkomen van redundantie.

Dus: voorkomen van redundantie is een het voorkomen van het dubbel vastleggen van gegevens.

Een goed datamodel bevat geen redundantie.

## Hoe maak je een ERD?

Elke opgave bestaat uit een "verhaaltje" dit wordt ook wel een case genoemd. Lees de case goed door en bepaal van welke entiteiten je gegevens wilt vastleggen. Bepaal van alle entiteiten *wat* je wilt vastleggen, dit zijn de attributen. Bepaal van alle attributen het datatype (integer, float, date, time, datetime of boolean). Vervolgens bepaal je de relatie tussen de entiteiten en bepaal je de primary keys en foreign keys.

Dus de stappen zijn:

1. Bepaal van welke entiteiten je gegevens wilt vastleggen in de database.
2. Bepaal van elke entiteit welke gegevens je wilt vastleggen, dit zijn de attributen en bepaal het datatype per attribuut.
3. Bepaal de relaties tussen de entiteiten
4. Bepaal de PK's en FK's.

De datamodellen worden gemaakt in [Lucichart](#), een datamodel kun je in PDF exporteren en op je eigen systeem bewaren.

[image-1603478193579.png](#)

Om een ERD in [Lucichart](#) te maken, maak je een gratis account en zoek je de ERD-template "Database ER Diagram", zie plaatje hierboven.

### Naming conventions

Het database design (het ERD) wordt in het Engels gemaakt. Entiteiten en Attributen worden via de [Snake Case](#) (in lower case) benoemd, bijvoorbeeld:

Entiteit: user

Attribuut 1: user\_name

Attribuut 2: user\_login\_count

Attribuut 3: last\_login\_date

Attribuut 4: password

Let ook op dat entiteiten **enkelvoud** zijn. Dus bijvoorbeeld *user* en niet *users*, of *leerlingen* niet *leerlingen*.

## Voorbeeld ERD

[image 1603486029048.png](#)

Dit diagram is in Lucichart gemaakt. Herken je de entiteiten en de attributen? Wat kun je van de relatie zeggen? Wat wordt er vastgelegd in deze database? Herken je de foreign key? Waar verwijst de foreign key naar?

## Opgave 1

Bestudeer het ERD

[image 1603486029048.png](#)

## Vraag 1

Hoeveel entiteiten zie je in het diagram?

- 1
- 2
- 3
- 8

## Vraag 2

Wat is de foreign key in dit ERD?

- id
- student\_id
- student
- sickness\_notification

## Vraag 3

De relatie die is getekend heeft een 'kraaienpootje' (of harkje). Het hakje staat aan de kant van de `sickness_notification`.

Wat betekent dit 'kraaienpootje'?

1. Elke student kan meer keer ziek zijn (geweest)
2. Elke ziektemelding kan door meer studenten worden gedaan.
3. Een ziektemelding hoort altijd bij één student.
4. Elk ID komt precies één keer voor en is dus uniek

## Vraag 4

In de database wordt de `birth_date` (=geboortedatum) vastgelegd. Je zou ook de leeftijd kunnen vastleggen?

Is dat een goed idee en waarom?

1. Ja goed idee om in plaats van `birth_day` de leeftijd vast te leggen. Leeftijd is namelijk kleiner en kost minder opslag ruimte.
2. Ja goed idee om beide vast te leggen. Je kunt dan heel snel de leeftijd of de geboortedatum (`birth_day`) opzoeken.
3. Nee geen goed idee om beide vast te leggen, kost veel te veel opslag ruimte.
4. Nee geen goed idee om leeftijd vast te leggen, dat verandert te vaak en je kunt de leeftijd altijd berekenen als je de `birth_day` weet.

## Vraag 5

Waarom normaliseer je (meerdere antwoorden mogelijk)?

1. Om dubbele opslag van gegevens te voorkomen.
2. Om te voorkomen dat gegevens in tegenspraak zijn met elkaar.
3. Om gegevens eenvoudig te kunnen wijzigen.
4. Om ervoor te zorgen dat je heel snel gegevens uit een database kunt opzoeken.
5. Om te voorkomen dat je telkens updates moet draaien.

## Vraag 6 - Tandarts

Je wilt bijhouden welke patiënten bij de tandarts welke behandelingen hebben gekregen.

Je wilt vastleggen welke patiënt met welke behandeling door welke tandarts is behandeld en wanneer dat is gebeurd.

Je wilt ook van een patiënt kunnen zien welke behandelingen bij hem of haar zijn uitgevoerd.

Je moet er rekening mee houden dat patiënten van tandarts kunnen wisselen omdat zij bijvoorbeeld verhuizen.

Elke behandeling heeft een prijs die je wilt vastleggen in de database.

Vink de entiteiten aan:

1. Tandarts
2. Patiënt
3. Behandeling
4. Lijst
5. Behandelprijs
6. Behandeldatum
7. Verhuizingen

--

---

Revision #11

Created 23 September 2021 08:14:55 by Max

Updated 28 September 2021 08:17:42 by Max