Normaliseren

Informatie

Wat is informatie en waaruit het bestaat?

Stel op een kaart staat *het cijfer 37* geschreven. Wat kun je dan zeggen van het cijfer 37? Niets bijzonders, toch? Alleen dat het een getal is, maar je weet niet wat het betekend of waar het een waarde van is.
Gaat het om een patiëntenkaart dan geeft 37 de lichaamstemperatuur van de patiënt weer.
Als op de kaart staat gefeliciteerd met je 37ste verjaardag. Wat geeft de 37 nu aan? In deze situatie is het de leeftijd van een persoon.

We hebben in *twee verschillende* situaties naar het getal 37 gekeken. Op een patiëntenkaart en op een verjaardagskaart. In beide situaties had het getal 37 een andere betekenis.

We kunnen concluderen uit het bovenstaand voorbeeld dat het getal 37 alleen een **feit** is. Onder feiten verstaan we namen, concrete voorwerpen, toestanden etc.

De *betekenis* van het getal 37 is in de eerste situatie de lichaamstemperatuur en in de tweede betekent het de leeftijd. De betekenissen van een feit noemt men het **gegeven**.

De twee situaties worden de **context** genoemd.

Informatie bestaat dus uit drie elementen, feiten, gegevens en een context. Waarbij gezegd kan worden dat: ***"Gegevens zijn de betekenissen van feiten in één bepaalde context."***

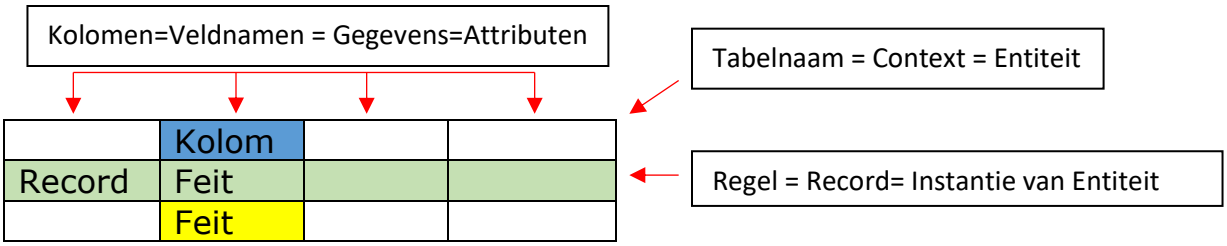
Over informatie kunnen we dus het volgende zeggen:

- Informatie bestaat uit feiten, gegevens en een context.
- Gegevens zijn de betekenissen van feiten in één bepaalde context.

Normaliseren

Informatieopslag

In computers en andere digitale apparatuur worden gebruikt om informatie op te slaan. De manier waarop dit gebeurt is met tabellen.



In de afbeelding hierboven zie je hoe de drie onderdelen van informatie worden toegepast op een tabel. De dingen die worden ingevuld in de hokjes zijn de feiten, de *(veld)*namen die we aan elke kolom geven zijn de gegevens. In database termen worden gegevens **attributen** genoemd. de naam van de tabel bepaalt de context. De tabel(naam) wordt een **entiteit** genoemd. Eén regel *unieke* informatie in een tabel is een **record** ofwel *een instantie* van de entiteit.

Binnen de wereld van automatisering wordt voor informatie het begrip **Data** gebruikt. Data zij dus eigenlijk alle feiten bij elkaar. Door dat we de feiten in een context plaatsen weet men wat er bedoeld wordt met de feiten.

Database

Programma's waar feiten in worden opgeslagen noemt men een ***database***. **Database Management Systemen** (DBMS) zijn applicaties waarmee informatie, ofwel data, verwerkt wordt door middel van computers. Databases komen tegenwoordig overal voor in mobiele telefoons (contactlijsten), in videospellen, administraties van scholen, banken, doctoren, etc.

Voordelen van het werken met database programma's zijn:

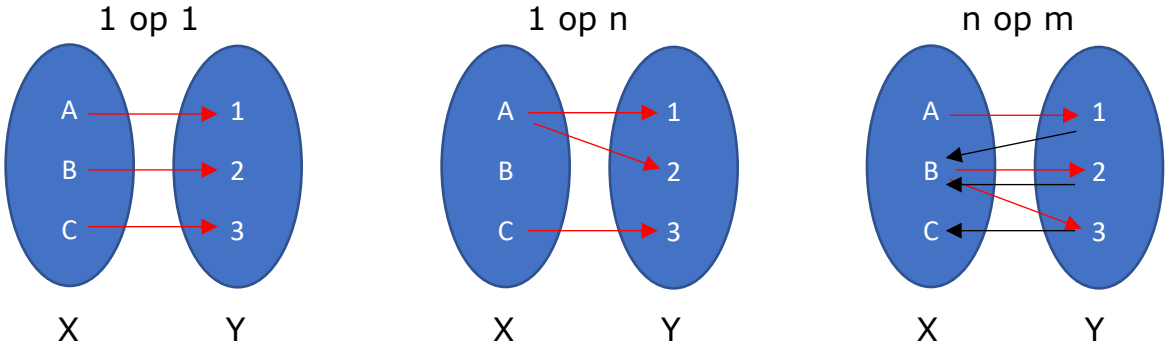
- Grote hoeveelheden data kunnen worden opgeslagen en worden beheerd met deze applicaties.
- Opvragen, zoeken en muteren (aanpassen of verwijderen) van informatie is snel, moet eenvoudig en nauwkeurig gedaan worden.
- Informatie is op alle mogelijke manieren te ordenen.
- Er kunnen op relatief eenvoudige wijze selecties worden gemaakt uit de data.

Normaliseren

Waarom normaliseren?

Normaliseren is een techniek die gebruikt wordt om data op een efficiënte manier in een **Relational Database Managment System** (RDBMS) op te slaan.

Dit gebeurt door de informatie te verdelen over meerdere tabellen die met elkaar gekoppeld worden, ***in relatie staan met elkaar***. Er worden drie type relaties onderscheiden:



1 op 1 relatie, *1 : 1 (één op één)***:**
Elk element in verzameling X heeft één of geen koppeling met één element in verzameling Y. Of wel elk element in verzameling Y is afhankelijk van een element in verzameling X.

Eén docent is mentor van één klas is een voorbeeld van een 1:1 relatie.

1 op n relatie, *1 : n (één op n=veel=∞, één op veel)***:**
Elk element in verzameling X heeft nul, één of meerdere koppelingen met verschillende elementen in verzameling Y.

Eén leerling heeft één of meerdere cijfers voor een vak is een voorbeeld van een 1:n relatie.

n op m relatie, n : m (*n op m, veel op veel*)**:**
Elk element in verzameling X heeft nul, één of meerdere koppelingen met verschillende elementen in verzameling Y. Tevens heeft elk element in verzameling Y nul, één of meerdere koppelingen met verschillende elementen in verzameling X.

Meerdere leerlingen volgen meerdere vakken en meerdere vakken worden gevolgd door meerdere leerlingen is een voorbeeld van een n:m relatie.

Normaliseren

Er is nog een **optionaliteit**. Soms is het ook mogelijk dat er géén relatie is. Er zijn dan bijvoorbeeld 0, 1 of meerdere relaties mogelijk. 0,1 : n (nul, één of meer).

Nul, één of meerdere leerlingen waren afwezig voor een proefwerk op dd/mm/jj is een voorbeeld van een 0,1:n relatie.

Normaliseerproces

Het normaliseerproces bestaat in principe uit vijf fasen, maar van de vijf wordt meestal alleen de eerste drie uitgevoerd.

1. De eerste regel bestaat uit drie deel stappen:
   A. ***Herhalende groep*** gegevens in aparte tabellen onderbrengen en voorzien van een ***primaire sleutel***.
   B. ***Proces gegevens*** verwijderen uit de tabel.
   C. ***Samengestelde gegevens*** splitsen in aparte atomaire gegevens (*kolomen/velden*).
2. Kolommen afhankelijk maken van de *gehele* primaire sleutel.
3. Kolommen onafhankelijk van elkaar maken en *alleen* afhankelijk van de primaire sleutel.

Herhalende groep zijn gegevens die meerdere keren voorkomen in een tabel. Stel dat we alleen een tabel cijfers van leerlingen zouden bijhouden dan zou de tabel er als volgt uit kunnen zien:

VN	AN	Adres	Tel/Cel	Vak	Cijfer	Datum
Pedro	Maduro	Rooi Afo 16b	5673131	NE	6.7	14/2/09
Pedro	Maduro	Rooi Afo 16b	5673131	WA	7.9	14/2/09
Pedro	Maduro	Rooi Afo 16b	5673131	EN	6.3	1/4/09
Maria	Ras	Tarabana 19a	7346679	NE	8.1	14/2/09
Maria	Ras	Tarabana 19a	7346679	SP	9.5	7/3/09

Je ziet dat de feiten van elke leerling herhaald moeten worden per cijfer. Maken we echter een tabel voor de leerlingen en een tabel voor de cijfers dan worden een aantal feiten niet meer dubbel opgeslagen.

Tabel Leerling

STAM	VN	AN	Adres	Tel/Cel
10000	Pedro	Maduro	Rooi Afo 16b	5673131
10001	Maria	Ras	Tarabana 19a	7346679

Normaliseren

Tabel Cijfers

STAM	Vak	Cijfer	Datum
10000	NE	6.7	14/2/09
10000	WA	7.9	14/2/09
10000	EN	6.3	1/4/09
10001	NE	8.1	14/2/09
10001	SP	9.5	7/3/09

Door gebruik te maken van het stamboeknummer is nog steeds te achterhalen welk cijfer hoort bij welke leerling.

Primaire sleutels zijn gegevens die **uniek** zijn voor een specifieke verzameling feiten. Ze definiëren één regel in een tabel uniek. Zo'n regel wordt in database termen een ***record*** genoemd. Vaak worden hiervoor codes of nummers gebruikt.
Denk aan het stamboeknummer van een leerling op school, het paspoortnummer van elke paspoort, het persoonsnummer op Aruba.

Samengestelde gegevens zijn gegevens die eigenlijk uit meerdere gegevens zijn opgebouwd. Bijvoorbeeld het adres. Deze bestaat op Aruba uit Stad, Straatnaam/wijknaam (*bario naam*) en het huisnummer. Soms is het echter niet noodzakelijk om een samengesteld gegeven verder op te splitsen. Een geboortedatum hoeft (*meestal*) niet verder te worden opgesplitst in *dag*, *maand* en *jaar*.

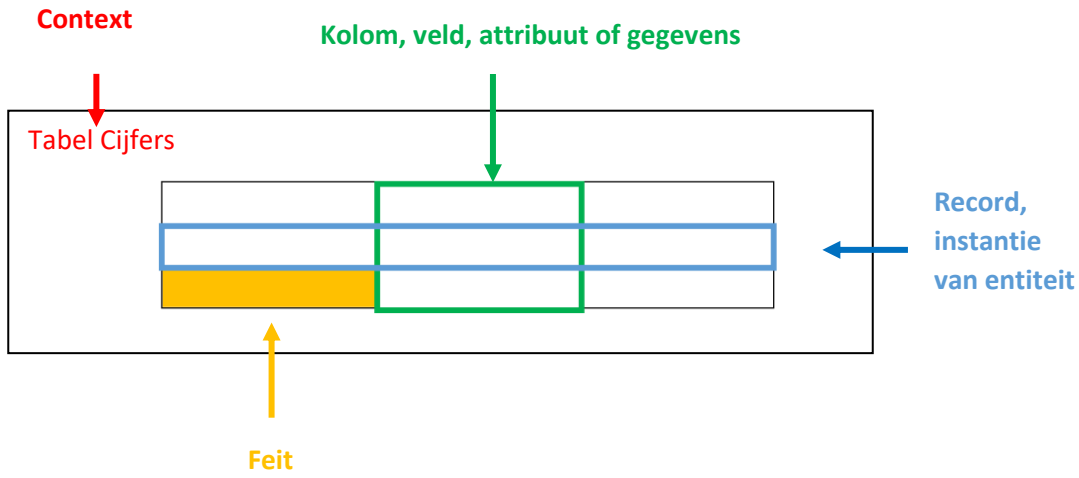
De gegevens die niet verder kunnen worden opgesplitst heten ***atomaire gegevens***. Zo is een dag een dag en deze is niet verder te verdelen in andere gegevens.

Proces gegevens bevatten informatie die berekent moet worden. Deze worden niet opgenomen in de tabellen van een database. Ze worden iedere keer opnieuw berekend om zo de meest actuele waardes te bevatten.

Normaliseren

Opmerking:
De termen *velden*, *gegevens*, *attributen* en *kolommen* worden soms door elkaar gebruikt maar zijn eigenlijk allemaal hetzelfde.

De onderstaande afbeelding geeft de termen weer van informatie in relatie tot een tabel zoals die gebruikt wordt in een database.



Door te normaliseren:

- wordt zoveel mogelijk het opslaan van dubbel informatie in de database bestreden.
- verwijderen van informatie ***uit alle*** tabellen van de database gebeurd automatisch.
- aanpassen van informatie in alle tabellen van de database gebeurd automatisch.
- zorgt dat de informatie consistent blijft

Normalisatie voorbeeld

De bibliothecaris van de school wil een simpel programma hebben om bij te houden welke leerling welk boek heeft geleend, wanneer, wanneer deze terug moet worden gebracht en wat het eventuele boete bedrag is dat deze leerling moet betalen wanneer het boek te laat wordt terug gebracht.
Van elke leerlingen moet bekend zijn het stamboeknummer, de voor- en achternaam, email, cel en huidige klas.

Van een boek moet bekend zijn de titel, de schrijver, het genre (= *soort boek*), de druk, de uitgever en het ISBN.

Normaliseren

Nulde normaalvorm (0de NV)

Als eerst moet de nulde normaalvorm (**0NV**) bepaald worden. Dit is een tabel waarin alle *gegevens* die nodig zijn bij elkaar worden gebracht. Deze tabel wordt ook wel een ***flat table*** (*platte tabel*) genoemd.

Door gebruik te maken van de eerder gegeven informatie ziet de 0NV tabel er als volgt uit:

Uitleen_tbl(boektitel, schrijver_naam, genre, druk, uitgever, ISBN, stamboeknr, voornaam (VN), achternaam (AN), email, cel, klas, leendatum, inleverdatum, boete bedrag)

Eerste normaalvorm (1ste NV)

Door de drie regels van de 1NV toe te passen,

A. Herhalende groepen gegevens in aparte tabellen onderbrengen en voorzien van een ***primaire sleutel***.
B. ***Proces gegevens*** verwijderen uit de tabel.
C. ***Samengestelde gegevens*** splitsen in aparte atomaire gegevens (*kolomen/velden*).

Ontstaan de volgende tabellen:

Uitleen_tbl(boekID, boektitel, schrijver_voornaam, schrijver_achternaam, genre, druk, uitgever, ISBN, stamboeknr, voornaam (VN), achternaam (AN), email, cel, klas)

geleend(boekID, stamboeknr, leendatum, inleverdatum)

De herhaalde groep gegevens zijn leendatum, inleverdatum, stamboeknr en boekID. Elk boek kan namelijk meerdere keren geleend worden door dezelfde of andere leerlingen. En omdat er meerdere exemplaren van een boek zijn, die allemaal hetzelfde ISBN hebben wordt boekID toegevoegd om elk boek uniek te identificeren.

Het boete bedrag is een gegeven dat berekend moet worden. Deze wordt dus niet opgenomen in de tabellen. Iedere keer dat het gegeven noodzakelijk is wordt het opnieuw berekend. Zo wordt altijd de meest up-to-date waarde aan de gebruiker medegedeeld.

De datums zijn samengestelde gegevens maar voor dit doel hoeven deze niet te worden opgesplitst in dag, maand en jaar. Ook schrijversnaam is een

samengesteld gegeven dat minimaal bestaat uit voornaam en achternaam. Je zou ook tussenvoegsels en titel kunnen toevoegen, maar voor dit voorbeeld niet noodzakelijk.

De twee tabellen staan nu in 1NV.

Tweede normaal vorm (2NV)

Voor de 2NV moet de volgende regel worden toegepast op de tabellen die nu in 1NV staan. De regel van de 2NV was:

Kolommen afhankelijk maken van de gehele primaire sleutel.

In de tabel geleend(boekID, stamboeknr, leendatum, inleverdatum) zijn beide (leendatum en inleverdatum) beide afhangelijk van de gehele primaire sleutel. Het boek met boeKID is op die specifieke datum geleend aan de leerling met dat stamboeknummer. Het is op die datum door deze leerling weer ingeleverd.

Er veranderd dus niks aan deze tabel.

Voor de tabel,

Uitleen_tbl(boekID, boektitel, schrijver_voornaam, schrijver_achternaam, genre, druk, uitgever, ISBN, stamboeknr, voornaam (VN), achternaam (AN), email, cel, klas)

moet nu één voor een de velden van de tabel bekeken worden of deze afhankelijk zijn van de primaire sleutel (boekID) van deze tabel. Mocht dit niet zo zijn dan moet er een nieuwe tabel gemaakt worden die voor zien wordt van zijn eigen primare sleutel.

De velden klas, cel, email, AN, VN zijn niet afhankelijk van boekID, maar alleen van stamboeknr. Dus deze moeten worden verwijderd en in een andere tabel worden onder gebracht.

Je krijgt nu de volgende tabellen:

Uitleen_tbl(boekID, boektitel, schrijver_voornaam, schrijver_achternaam, genre, druk, uitgever, ISBN, <*stamboeknr*>)

Leerling(stamboeknr, voornaam (VN), achternaam (AN), email, cel, klas)

geleend(boekID, stamboeknr, leendatum, inleverdatum)

Normaliseren

Het attribuut <*stamboeknr*> in de Uitleen_tbl zorgt voor de relatie met de tabel Leerling. Het stamboeknr is de *primaire sleutel* in de tabel leerling en wordt toegevoegd in de Uitleen_tbl. Dit attribuut wordt daarom ook wel de **vreemde sleutel** ook wel secundaire sleutel of secondary key genoemd.

Deze drie tabellen staan nu dus in de 2de normaalvorm.

Derde normaal vorm (3NV)

Om de tabellen van de 2NV in de 3NV te zetten moet de volgende regel worden toegepast op die van de 2NV:

Kolommen onafhankelijk van elkaar maken en alleen afhankelijk van de primaire sleutel.

Men moet deze regel nu toepassen op de tabellen die nu in 2NV staan.

Bij de tabel uitleen,

Uitleen_tbl(boekID, boektitel, schrijver_voornaam, schrijver_achternaam, genre, druk, uitgever, ISBN, <*stamboeknr*>)

Is dus de schrijver_voornaam en schrijver_achternaam, zijn niet afhankelijk van de primaire-sleutel (boekID) en moet dus in een eigen tabel geplaast worden. Dit levert nu de volgende tabellen op:

Boek(boekID, boektitel, <schrijverID>, genre, druk, uitgever, ISBN)
Schrijver(SchrijverID, voornaam, tussenvoegsels, achternaam, titel)
Leerling(stamboeknr, voornaam (VN), achternaam (AN), email, cel, klas)
geleend(<boekID>, <stamboeknr>, leendatum, inleverdatum)

Deze tabellen staan nu in de 3NV.

Opmerking:
Als voor de uigever meer dan alleen de naam van de uitgever zou worden bijgehouden dan zou ook voor de uitgever een aparte tabel moeten worden gemaakt. Deze zou er als volgt uitkunnen zien:

Uitgever(uitgeverID, naam, staatnaam, huisnummer, postzip-code, tel, email, url-website)

Het attribuut uigeverID zou dan uitgever moeten vervangen in de tabel Boek als secundaire sleutel.

Normaliseren

Datadictionary

Om de tabellen aan te maken in een RDBMS is het nodig dat bekend is van welk type elke van de velden in de tabel is en welke velden een primary of secondary key zijn.

Dit wordt in een tabel weergegeven die een data dictionary wordt genoemd. Hieronder zie je een voorbeeld van hoe zo'n data dictionary er uit zo kunnen zien voor de tabel Boek(boekID, boektitel, <schrijverID>, genre, druk, uitgever, ISBN).

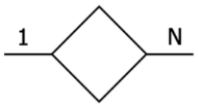
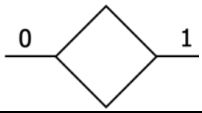
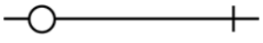
Attribuut	Datatype	
boekID	Varchar(5)	Not null, primary key
boektitel	Varchar(30)	
schrijverID	Integer	Foreign key
genre	Varchar(15)	
Druk	Varchar(15)	
uitgever	Varchar(30)	
ISBN	Varachar(15)	

Met behulp van deze tabel is dan snel de code te schrijven om de tabel in een RDBMS aan te maken.

Probeer dit nu ook te doen voor de overige tabellen.

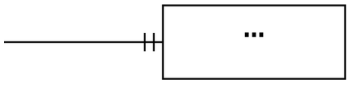
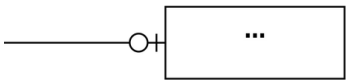
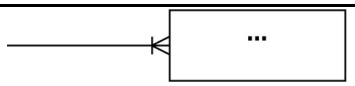
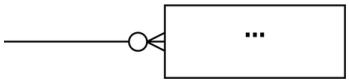
Bachmann en Entity-Relationship-Diagram

De tabellen en de relaties tussen de tabellen wordt grafisch weergegeven met een **Bachmann diagram** of **Entity-Relationship-Diagram** (ERD). De symbolen die gebruikt worden voor ERD zijn als volgt:

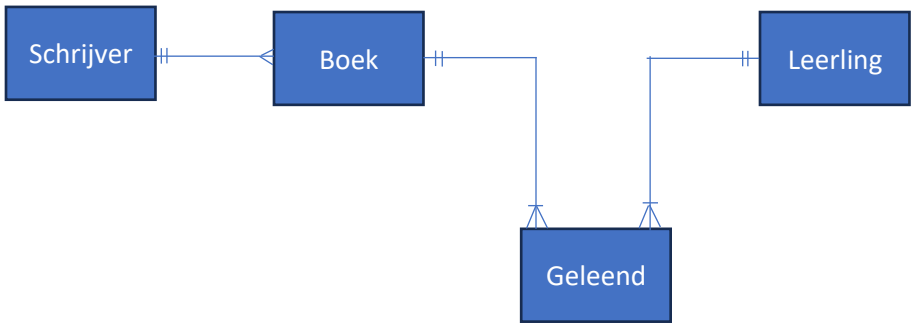
Onderdeel	Chen (ERD)	BachMann
Entiteit		
Relatie		
Cardinaliteit (één / veel)		
Optionaliteit (optioneel / verplicht)		

Normaliseren

De vier mogelijke combinaties van symbolen voor een Bachmann diagram:

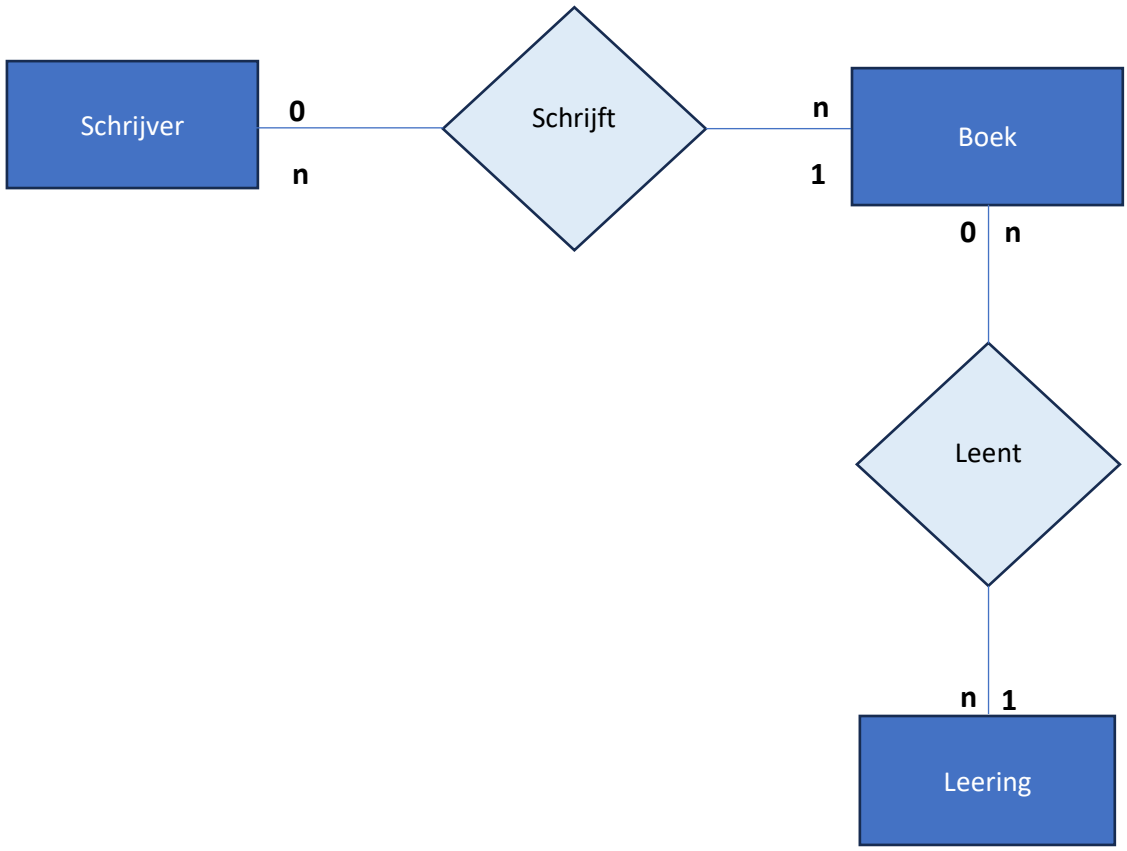
Optionaliteit	Cardinaliteit	Relatie type	Symbolisch
één	één	één en slechts één	
nul	één	nul of één	
één	meer	één of meer	
nul	meer	nul of meer	

Voor het boeken leensysteem zou de Bachmann diagram er zo uit zien:



Normaliseren

Het ERD zou er zo uit zien:



Een schijver schrijft 0,1 of meer boeken.
Een boek is geschreven door 1 of meerdere schrijvers

Een boek word 0,1 of meer keren geleent door leerlingen.
Een leerling leent 0, 1 of meerdere boeken