MODULE DATABASES
CSG AUGUSTINUS

GEGEVENSBANKEN, SQL & DATASTRUCTUREN

INHOUDSOPGAVE

Inhoud

INLEIDING

Ken je iemand wiens bureau eruit ziet zoals in figuur 1? De berg papieren bevat een hoop gegevens of **data**, maar ze leveren de eigenaar van het bureau op dit moment misschien niet de gewenste **informatie**.

FIGUUR 1

De noodzaak om data te structureren wordt groter, naarmate je meer data hebt. Dat geldt in jouw eigen omgeving, maar ook in de digitale wereld. Wie de data niet op een herkenbare manier ordent, heeft geen overzicht. Je kunt er dan geen **informatie** uithalen of nieuwe **kennis** mee vergaren.

Als je de data wel ordent, kun je er **informatie** uithalen: de data krijgt betekenis. Als je gegevens met elkaar weet te verbinden of er patronen in ontdekt, ontwikkel je nieuwe **kennis**. Zo levert één weerstation **data** (in de vorm van de spanning van de weersensoren) over de lokale weersituatie die wordt omgezet naar **informatie** over bijvoorbeeld windkracht, temperatuur en luchtdruk. Meerdere weerstations leveren een totaalbeeld voor een land, dat kan bijdragen aan de **kennis** in de vorm van een weerbericht.

Bij de lessen over programmeren heb je kennis gemaakt met het begrip **variabele**. In een variabele kan een gegeven worden opgeslagen onder een bepaalde naam, zoals *huisnummer = 4*. Een bijzondere variabele was de **array**, waarin je een reeks met gegevens onder één naam kon bewaren. Een voorbeeld hiervan is: *onderwerpVanHetHoofdstuk = ['html','javascript,'hardware','databases']*.
Als we dit idee verder uitbreiden komen we uit bij een **tabel**. Op de volgende pagina staan twee tabellen die in het vervolg van dit hoofdstuk vaker zullen worden gebruikt. De tabellen horen bij elkaar. We zeggen dat ze een **relatie** hebben. Een verzameling tabellen die een relatie met elkaar hebben, heet een **relationele database**. Het werken met databases is het hoofdonderwerp van deze module.

Opdracht 1: data, informatie en kennis

1. Wat is het verschil tussen data en informatie? Gebruik de theorie hierboven en internet.
2. In de theorie wordt gesproken over data, informatie en kennis. In deze volgorde neemt de waarde van de gegevens steeds verder toe. Zoek op internet op wat de vierde stap is.
3. Om een voorspelling van het weer van volgende week te kunnen doen, wordt een grote hoeveelheid data, informatie en kennis gecombineerd.
   Bedenk twee andere voorbeelden waarbij dit ook het geval is.
4. *Big data* is een actueel item binnen de ICT. Zoek op wat de term betekent.
5. Noem drie bedrijven die gegevens van jou hebben.
6. Bedenk een bedrijf dat iets zou kunnen hebben aan die gegevens. Leg uit waar ze de gegevens voor kunnen gebruiken.
7. Stel dat je een ruim budget hebt om gegevens te kopen. Welke gegevens zou je graag willen hebben? En waarom?

Opdracht 2: informatie uit tabellen halen

Bekijk de tekst op de pagina hiernaast en bestudeer de tabellen. Beantwoord vervolgens de vragen.

8. Van welk type meter bestaan de meeste varianten in de tabel?
9. Welk type meter is het vaakst vervangen? Hoe vaak?
10. Geef het type en het merk van de meter die op dit moment het meest in gebruik is bij de weerstations.
11. Welke meter werd het eerst van allemaal vervangen? En welke werd het snelst vervangen? Geef het type en het merk en het weerstation waar deze vervanging plaats had.
12. Hoeveel meters die werken op een spanning van 12V worden op dit moment nog gebruikt?
13. Welke beheerders maken op dit moment gebruik van meters die werken op 3V?
14. Bedenk drie vragen die je over deze tabellen zou kunnen stellen. Laat je buurman antwoorden.

Database Weerstations

Hiernaast zie je een tabel met samenwerkende weerstations in de provincie Groningen. Deze tabel **stations** spreekt waarschijnlijk voor zich: hierin staan de verschillende weerstations op een rij met de plaats waar ze zich bevinden en de persoon die het lokale weerstation beheert.

stations		
ws_id	plaats	beheerder
1	Zuidhorn	Van der Veen
2	Groningen	Velthuizen
3	Groningen	Grgurina
4	Bedum	Palsma
5	Garrelsweer	Osinga

FIGUUR 2

Een weerstation bevat verschillende **meters**. Er is een tabel met gegevens over verschillende typen meters. Hierbij staat ook vermeld van welk merk ze zijn en op welke spanning (voeding) ze werken. Voor elk type meter is er een unieke identificatiecode (*m_id*) gemaakt.

Niet elk weerstation zet dezelfde meters in. Beheerders maken zelf een keuze. Bovendien zijn in de loop der jaren meters kapot gegaan en vervangen door een ander type. Informatie hierover is te vinden in de tabel **inzet**.
Van een bepaald type meter kunnen meerdere exemplaren worden ingezet. Alle exemplaren hebben hun eigen metercode (*m_code*). Dit betekent dat bij één *m_id* meerdere metercodes (*m_code*) kunnen horen. Wanneer een meter is vervangen, is dat te zien doordat in de tabel het veld *eind* is gevuld. In s het veld niet gevuld, dan krijgt deze in de database de waarde **NULL**. De kolom *start* geeft de plaatsingsdatum van een nieuwe meter.

De tabel **inzet** is het lastigst te lezen en te begrijpen. De eerste regel met metercode (*m_code*) 3681 heeft 3 als *ws_id*. Dit is een identificatiecode van het weerstation van beheerder Grgurina uit de tabel **weerstations**.
De meter met de unieke code 3681 heeft metercode (*m_code*) 2. Dit betekent dat het een luchtdrukmeter van het merk Samsung is met een voeding van 12V. Dit blijkt uit de tabel **meters**.

Uit dit voorbeeld blijkt dat deze drie tabellen een relatie met elkaar hebben. Samen vormen de drie tabellen een gegevensbank of **relationele database**.

meters			
m_id	type	merk	voeding
1	luchtdruk	Samsung	5V
2	luchtdruk	Samsung	12V
3	luchtdruk	Philips	5V
4	luchtdruk	Philips	5V
5	luchtdruk	Vavetech	5V
6	luchtdruk	Vavetech	12V
7	luchtdruk	Vavetech	5V
8	windkracht	Vavetech	3V
9	windkracht	Samsung	5V
10	windkracht	Samsung	5V
11	windkracht	Samsung	3V
12	temperatuur	Vavetech	5V
13	temperatuur	Philips	5V
14	temperatuur	Philips	5V
15	temperatuur	Samsung	5V
16	temperatuur	Samsung	12V
17	temperatuur	Samsung	3V
18	temperatuur	Samsung	3V

inzet				
m_code	ws_id	m_id	start	eind
3681	3	2	4-10-2005	28-9-2008
3226	3	8	4-10-2005	
3847	3	13	4-10-2005	7-1-2009
3412	4	2	4-10-2005	9-9-2007
3924	5	2	4-10-2005	30-10-2008
3922	5	12	4-10-2005	
3918	1	5	14-7-2007	
3582	1	7	14-7-2007	
3637	1	12	14-7-2007	
3512	4	4	9-9-2007	20-12-2011
3134	3	6	28-9-2008	
3080	3	12	7-1-2009	
3338	5	15	1-1-2010	1-6-2010
3913	5	8	1-1-2010	
3356	2	9	27-3-2016	
3607	4	5	20-12-2011	
3721	2	1	27-3-2016	
3251	2	15	27-3-2016	
3664	5	7	14-7-2017	

FIGUUR 3

H1 SQL ZOEK EN VIND

1.1 SQL: de computer laten zoeken

Bij opdracht 2 moet je zelf zoeken in de drie tabellen van de database met gegevens van weerstations. Dat wordt al snel een vervelend klusje. Bedenk hoe langdurig jouw zoektocht naar de antwoorden zou zijn geweest, als alle weerstations van Nederland in de tabel waren opgenomen. En als we voor al die weerstations nog meer meetapparatuur zoals regenmeters, luchtvochtigheidsmeters etc. hadden toegevoegd!

Wanneer vraagstukken niet meer te overzien zijn voor mensen of je veel sneller antwoorden wilt krijgen, kun je gebruik maken van de computer. Maar hoe stel je een vraag aan de computer? Hoe leg je de computer uit naar welke gegevens je op zoek bent?
Hiervoor is een speciale taal ontwikkelt: **SQL** of *Structured Query Language*. Met SQL kun je niet alleen een database bevragen om gegevens te selecteren. Je kunt ook gegevens toevoegen, wijzigen of verwijderen en berekeningen uitvoeren, sorteren, groeperen etc. In dit hoofdstuk leer je een aantal van deze technieken om rijen uit een tabel te selecteren. Zo'n rij heet een **record**. Hieronder staan een aantal voorbeelden van sql-zoekopdrachten of **query's**.

```
SELECT * FROM stations                                  -- Toont de tabel stations
SELECT merk FROM meters                                 -- Toont de kolom merk van de tabel meters
SELECT merk,voeding FROM meters                         -- Toont zowel de kolom merk als voeding
SELECT beheerder FROM stations WHERE plaats='Groningen' -- Namen van beheerders in Groningen
```

In bovenstaande voorbeelden staan typische SQL-uitdrukkingen. Ze zijn hier voor de duidelijkheid met hoofdletters geschreven, maar dat hoeft niet perse. Met **SELECT** geef je aan dat je gegevens wilt selecteren, met **FROM** geef je aan in welke tabel of tabellen de computer moet zoeken. Achter SELECT kun je aangeven welke kolommen (informatie) je terug wilt krijgen. Met de asterisk (*) geef je aan dat je alle kolommen wilt selecteren.

Met **WHERE** kun je een voorwaarde of eis toevoegen. In de laatste query hierboven wordt een lijst met beheerders opgevraagd met de eis dat ze een weerstation in Groningen moeten beheren. Is er meer dan één voorwaarde? Gebruik **AND**: SELECT * FROM meters WHERE type='luchtdruk' AND voeding='5V'

SQL leer je het beste door het te oefenen. Van je leraar krijg je een omgeving met meerdere databases die je kunt gebruiken bij de opdrachten in deze module.

Noteer alle query's en overige antwoorden in een schrift of in een Word-document! Wanneer gevraagd wordt om iets te selecteren, noteer je de query. Wordt er een andere vraag gesteld waarvoor je een query moet gebruiken om het antwoord te vinden, dan noteer je zowel de query als het antwoord.

Opdracht 3: Oefenen met selecteren

Gebruik de database *weerstations*.

15. Voer de query's uit de theorie uit en bekijk het resultaat.
16. Selecteer alle (gegevens van) meters met een 3V-voeding.
17. Welke temperatuurmeter van het merk Samsung werkt op een 5V-voeding? Geef alleen de *m_id*.
18. Gebruik de *m_id* uit de vorige vraag om uit te zoeken hoeveel van deze meters zijn ingezet. Geef de codes (*m_code*), startdatum en einddatum van deze meters.
19. Selecteer alle (gegevens van) luchtdrukmeters van het merk Vavetech met een 5V-voeding.

Opdracht 4: Kennismaken met de database Top-2000

De top-2000 is een jaarlijks terugkerend radioprogramma. Tussen Kerst en Nieuwjaar worden op radio 2 de beste 2000 liedjes aller tijden afgespeeld. Wat die beste liedjes zijn wordt bepaald door mensen die online hun voorkeur aangeven.
Gebruik de database *Top 2000 v1*. Er is ook een database *Top 2000 v2*. Daar gaan we verderop in deze module mee aan de slag. In de database staan de *titel* van het nummer en de naam van de *artiest*. De kolom *uit* is het jaar waarin het nummer is gemaakt. De kolommen *ed2017*, *ed2018*, *ed2019*, *ed2020* en *ed2021* geven de positie van het nummer in de editie van de top 2000 van dat kalenderjaar. De waarde NULL (leeg veld) in één van deze kolommen betekent dat het nummer dat jaar geen notering in de top-2000 had.

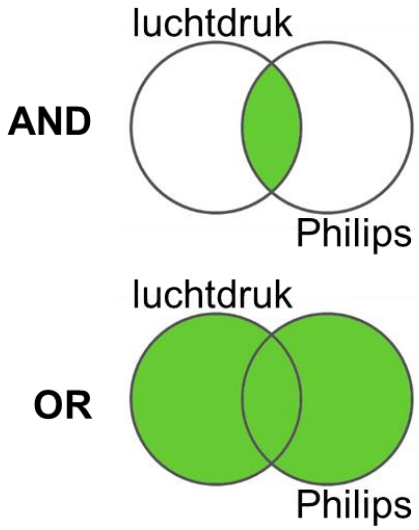
20. Voer de query *SELECT * FROM top2000* en bekijk de inhoud van de database.
21. Selecteer het record van het nummer dat op positie 1627 stond in de top-2000 van het jaar 2021?
22. In welk jaar had Maan voor het eerst een notering in de top-2000?
    Selecteer alle records met de naam van de artiest om tot je antwoord te komen.
23. Hoeveel nummers van Adele die gemaakt zijn in 2011 hebben een notering gehad in de top-2000?
    Selecteer alleen de titel en de kolom *uit*.
24. Welke artiesten hebben een notering gehad in de top-2000 met de titel *Sorry*, maar stonden niet in de editie van 2020? Zoek voor het beantwoorden van de vraag alle liedjes met de titel *Sorry*.
25. Zoek de noteringen van je favoriete artiest en of nummer op.

1.2 Logische operatoren

In paragraaf 1.1 hebben we gezien dat we twee voorwaarden kunnen combineren met de uitdrukking **AND**. Dit commando gebruikt je wanneer aan beide voorwaarden tegelijkertijd moet worden voldaan. De Venndiagrammen in figuur 4 tonen het verschil tussen **AND** en **OR**.

De linker cirkel vertegenwoordigt een verzameling meters van het type luchtdruk en de rechter cirkel vertegenwoordigt alle meters van het merk Philip. Het groen gemarkeerde deel van **AND**-tekening bevat dan alle luchtdrukmeters van het merk Philips. Bij **OR** maakt het niet uit: het groen gemarkeerde deel (alles) bevat alle meters die of van het type luchtdruk zijn of van het merk Philips.

Een bijzondere operator is **NOT** wanneer je juist wilt dat aan een eis niet wordt voldaan. En zo zijn er nog veel meer operatoren die je kunt gebruiken. De meeste zullen je bekend voorkomen van de reken- of wiskundelessen en van de programmeerlessen, toen je aan de slag ging met voorwaarden (*if*) en herhalingen (*for* en *while*).
Hieronder zie je voorbeelden van een aantal andere operatoren:



FIGUUR 4

```
SELECT * FROM meters WHERE voeding='3V' OR voeding='12V'                    -- Q1
SELECT * FROM stations WHERE NOT plaats='Groningen'                         -- Q2
SELECT * FROM inzet WHERE m_code<3200                                       -- Q3
SELECT * FROM inzet WHERE NOT eind IS null                                  -- Q4
SELECT * FROM stations WHERE ws_id>=4                                       -- Q5
SELECT * FROM meters WHERE NOT (voeding='5V' AND NOT merk='Samsung')        -- Q6 Let op de haakjes!
```

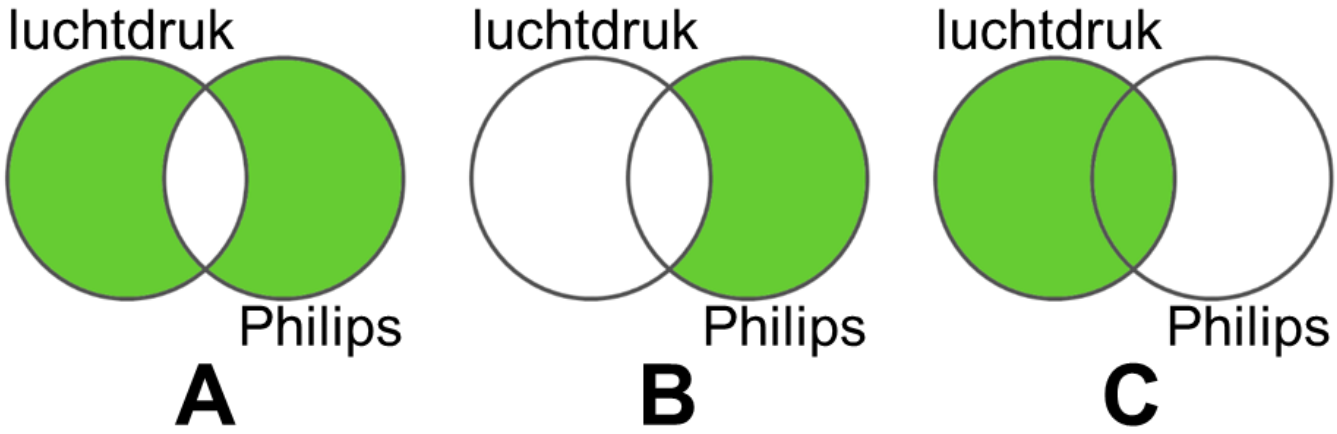
Opdracht 5: Logisch zoeken

Gebruik de database *weerstations*.

26. Lees de query's Q1 t/m Q6 uit de theorie op de vorige bladzijde en voorspel met behulp van de gegeven tabellen van de database weerstations wat het resultaat zal zijn.
27. Voer de query's nu uit en vergelijk jouw voorspelling met het resultaat. Begrijp je de query's?
28. Genereer een overzicht van alle windkrachtmeters met een een *m_id* die kleiner is dan 10.
29. Genereer een overzicht van alle temperatuurmeters die niet van Samsung zijn.

Extra opdracht 6: nog meer logisch zoeken

Gebruik de database *weerstations*.



FIGUUR 5

30. Bekijk de Venndiagrammen A, B en C in figuur 5. Beschrijf voor A, B en C in woorden welke meters er worden geselecteerd als de Venndiagrammen worden toegepast op de tabel *meters*.
31. Schrijf de query's die horen bij A, B en C.
32. Voer de query's uit. Controleer voor elke query of het resultaat overeen komt met jouw omschrijving en of het resultaat klopt met de gegeven tabel van *meters*.
33. Voorspel wat het resultaat is van de query:
    SELECT * FROM meters WHERE NOT (type='luchtdruk' AND merk='Philips')
    Controleer daarna je antwoord.
34. Voorspel wat het resultaat is van de query:
    SELECT * FROM meters WHERE NOT type='luchtdruk' AND NOT merk='Philips'
    Controleer daarna je antwoord.
35. Voorspel wat het resultaat is van de query:
    SELECT * FROM meters WHERE NOT (type='luchtdruk' OR merk='Philips')
    Controleer daarna je antwoord.

Opdracht 7: Logisch zoeken in de top-2000

Gebruik de database *Top 2000 v1*.

36. Maak een overzicht van de top-2000 van 2017 met achtereenvolgens de kolommen *ed2017*, *artiest* en *titel*. Let goed op: hoeveel records heeft jouw resultaat? (En hoeveel is logisch?)
37. Welk nummer van Oasis stond in de editie van 2017 in de top-100 van de top-2000?
38. Selecteer alle nummers van Anouk die gemaakt zijn na 2009.
39. Selecteer alle nummers van Anouk die tussen 2005 en 2010 uitkwamen (2005 en 2010 doen mee!).
40. Selecteer alle nummers uit de database van Maroon 5 en alle nummers van Stromae.
41. Welke artiesten hebben een liedje gemaakt dat voor 2000 uitkwam en de titel *One* had?
    Maak een overzicht met alleen de naam van de artiest en het jaar waarin het liedje is gemaakt.
42. (EXTRA) Hoeveel liedjes uit 1976 stonden in 2017 nog in de top-1500 van de top-2000 maar een jaar later buiten de top-1500? Genereer een overzicht van de records. (HINT 4 records is fout)

1.3 Resultaten ordenen & dubbel resultaat vermijden

We zijn inmiddels in staat om records uit een database te selecteren op basis van één of meerdere eisen. Nu willen we de records ook op volgorde kunnen leggen. Het **sorteren** van records doe je met de opdracht **ORDER BY**:

```
SELECT * FROM stations ORDER BY plaats
-- Zet de records op (alfabetische) volgorde van plaats

SELECT * FROM stations ORDER BY plaats ASC
-- Sorteer van a t/m z (hetzelfde resultaat als hierboven)

SELECT * FROM stations ORDER BY plaats DESC
-- Sorteer van z t/m a
```

Het resultaat van de query's zie je in figuur 6. De eerste twee query's geven het resultaat uit de bovenste tabel, de laatste query geeft het resultaat uit de onderste tabel.

Met **ASC** (Engels: *ascending* = oplopend) en **DESC** (Engels: *descending* = aflopend) geef je aan op welke manier je wilt sorteren.

stations

ws_id	plaats	beheerder
4	Bedum	Palsma
5	Garrelsweer	Osinga
2	Groningen	Velthuizen
3	Groningen	Grgurina
1	Zuidhorn	Van der Veen

stations

ws_id	plaats	beheerder
1	Zuidhorn	Van der Veen
2	Groningen	Velthuizen
3	Groningen	Grgurina
5	Garrelsweer	Osinga
4	Bedum	Palsma

FIGUUR 6

Stel dat je wilt weten welke bedrijven allemaal meters maken voor onze weerstations. Je zou dan met een query de kolom *merk* uit de tabel *meters* kunnen selecteren, maar dan krijg je voor elke meter het merk. Het gevolg hiervan is dat je negen keer "Samsung" op je beeldscherm krijgt. Eén keer is in dit geval wel genoeg. De oplossing is het gebruik van **DISTINCT** (Engels: uniek). De computer geeft dan alleen een nieuwe bedrijfsnaam terug als die verschilt van de reeds vermelde namen:

```
SELECT DISTINCT merk FROM meters          -- Geeft unieke merknamen

SELECT DISTINCT merk,type FROM meters     -- Geef unieke combinaties van merk en type
```

De onderste query hierboven zoekt unieke combinaties van type en merk. Hierbij maakt het niet uit of de voeding verschilt!

Opdracht 8: ORDER BY en DISTINCT

Gebruik de database *weerstations*.

43. Geef een alfabetisch overzicht van beheerders. Zorg dat alleen de naam wordt getoond.
44. Sorteer de tabel *inzet* op basis van de startdatum waarop de meter is geplaatst.
45. Voer de volgende query uit en bekijk het resultaat. Wat is het verschil met de bovenste tabel uit figuur 6? Wat moet je van dit voorbeeld leren?
    SELECT * FROM stations ORDER BY plaats,beheerder
46. Voer de volgende twee query's uit. Leg uit wat het verschil is tussen de query's:
    A: SELECT * FROM meters ORDER BY type,merk
    B: SELECT * FROM meters ORDER BY merk,type
47. Wat verandert er allemaal aan het resultaat als we query B van de vorige vraag aanpassen tot:
    SELECT merk,type,voeding FROM meters ORDER BY merk DESC,type ASC,voeding DESC
    ? Probeer het antwoord te geven zonder de query uit te voeren.
48. Kijk nog eens goed naar het resultaat van de query uit de vorige vraag. Wat valt je op aan de sortering van de kolom voeding die gesorteerd is met DESC?
49. Maak een overzicht van de verschillende meters die in de database voorkomen. Elk type meter mag maar één keer worden genoemd.
50. Wat is het resultaat van de query: SELECT DISTINCT voeding,type FROM meters ORDER BY type

Opdracht 9: ORDER BY en DISTINCT in de top-2000

Gebruik de database *top 2000 v1*.

51. Toon alle door Michael Jackson gemaakte liedjes met een notering in de top 2000, op volgorde van het jaar van uitgave. Het resultaat moet alleen de titel en het jaar bevatten, met het meest recente nummer bovenaan.
52. De artiest Sting (figuur 7; midden) was vroeger de zanger van The Police. Geef een alfabetisch overzicht van alle nummers van The Police en Sting. Vermeld alleen de artiest en de titel.
53. Sorteer het resultaat van de vorige vraag op jaar van uitgave. Lukt dit ook zonder dat het jaar van uitgave wordt getoond in het resultaat?
54. Maak een alfabetische lijst van alle artiesten die een notering in de top-20 van de top-2000 van 2018 hadden. Elke artiestennaam mag maar één keer voorkomen.
    Hoeveel verschillende artiestennamen vind je in de lijst?
55. Er staan tien liedjes van Racoon in de database. Een aantal daarvan is in hetzelfde jaar gemaakt.
    Hoeveel verschillende jaren van uitgave zijn er voor liedjes van Racoon?
    Maak een aflopend gesorteerd overzicht van deze jaren van uitgave van liedjes van Racoon. Zorg dat elk jaar maar één keer voorkomt.

FIGUUR 7 THE POLICE MET STING

Extra opdracht 10: Als je het niet precies weet

Tot nu toe hebben we query's gemaakt waarbij we steeds de exacte zoekterm (zoals naam van de artiest) wisten. Maar wat nu als je het niet helemaal weet? Daarvoor gebruiken we de operator **LIKE** in combinatie met de **wildcard**-symbolen % en _.

Gebruik de database *top 2000 v1*.

56. Bekijk het resultaat van de query:
    SELECT DISTINCT artiest FROM top2000 WHERE artiest LIKE 'Av%'
    Wat is in dit geval de betekenis van het %-teken?
57. Hoeveel unieke artiestennamen die eindigen op een i staan er in de database, denk je? Doe eerst eens een gok en maak daarna een query die het overzicht van deze namen geeft (waarbij elke naam maar één keer in de lijst mag staan).
58. Leg zorgvuldig in woorden uit wat het resultaat is van de query:
    SELECT DISTINCT artiest FROM top2000 WHERE artiest LIKE 'The%Brothers'
    Controleer daarna jouw beschrijving. Was die zorgvuldig genoeg voor alle resultaten van de query?
59. Bekijk het resultaat van de query:
    SELECT DISTINCT artiest FROM top2000 WHERE artiest LIKE 'Tom _____'
    Wat is in dit geval de betekenis van het teken _? (De query bevat na de spatie vijf opeenvolgende _'s!)
60. Hoeveel liedjes met de term Christmas in de titel staan er in de database?
61. Leg uit wat het verschil is tussen de query's:
    A: SELECT artiest,titel FROM top2000 WHERE artiest LIKE 'the%'
    B: SELECT artiest,titel FROM top2000 WHERE artiest LIKE 'the %'
62. Zie de vorige vraag. Welke resultaten volgen niet uit de ene maar wel in de andere query? Bedenk een query die als laat zien welke artiesten wel uit query A maar niet uit query B volgen.
63. Hoeveel artiestennamen van drie letters zijn er?
64. Hoeveel vermeldingen zijn er waarbij zowel de naam van de artiest als de titel de letter q bevat?
65. Bedenk zelf een gekke vraag die je buurman / - vrouw moet beantwoorden met de LIKE-operator.

FIGUUR 8 TOM WAITS

1.4 Tellen, groeperen en rekenen

Hoeveel leerlingen zitten er in havo-4? Hoeveel krentenbroden hebben we nog op voorraad? Hoeveel lampen zijn er het afgelopen jaar vervangen?

FIGUUR 9 VOORRAADBEHEER

In veel systemen met een database spelen aantallen een grote rol. Ze leveren cruciale informatie en kennis op voor bedrijven en organisaties. Vaak staan die aantallen niet als getal in de database, maar worden de aantallen berekend met SQL. Als we willen tellen, gebruiken we **COUNT** (Engels: tel of tellen).

Hieronder staan twee voorbeelden van de toepassing van COUNT:

```
SELECT COUNT(*) FROM stations WHERE NOT plaats='Groningen'
-- Tel het aantal stations dat niet in de plaats Groningen ligt.

SELECT plaats,COUNT(*) AS hoeveelheid FROM stations GROUP BY plaats ORDER BY hoeveelheid DESC
-- Tel het aantal stations per plaats en geef de kolom met dat aantal de titel hoeveelheid. Sorteer op deze hoeveelheid
```

In figuur 10 zie je het resultaat van de tweede query hierboven. De tweede kolom in deze tabel heeft de naam *hoeveelheid* meegekregen. Dit is gedaan met **AS**. Het getelde aantal is geen kolom in de oorspronkelijke database. Daarom kun je er een **ALIAS** aan meegeven: *hoeveelheid* is hier de alias.

stations	
plaats	**hoeveelheid**
Groningen	2
Zuidhorn	1
Bedum	1
Garrelsweer	1

FIGUUR 10

De tweede query telt niet zomaar het totale aantal stations (de eerste query hierboven doet dat wel!). In plaats daarvan worden de weerstations per plaats gegroepeerd met **GROUP BY**. Het zijn dus de aantallen per plaats.

Natuurlijk kunnen we de computer niet alleen laten tellen. In de opdrachten leer je werken met bijvoorbeeld gemiddelde (**AVG** = *average*) en minimum (**MIN**).

Opdracht 11: Rekenen en groeperen

Gebruik de database *weerstations*.

66. Schrijf een query die het aantal meters met een 3V-voeding telt.
67. Genereer een tabel met de kolommen *merk* en *aantal* die het aantal verschillende types meters per merk in de tabel meters telt. Zorg dat het merk met de meeste verschillende meters bovenaan staat.
68. Voorspel wat de volgende query doet en controleer daarna het resultaat:
    SELECT MAX(m_code),start FROM inzet
69. Uit de tabel *inzet* kan worden gehaald hoe vaak een bepaald type meter (*m_id*) is ingezet. Welke meter is het vaakst ingezet of ingezet geweest?
    Maak een overzicht van *m_id* en dit aantal, op volgorde van het aantal van hoog naar laag. Deze kolom geef je de naam *N*.

Opdracht 12: Rekenen aan de top-2000 database

Gebruik de database *top 2000 v1*.

70. Maak een query die telt hoeveel liedjes van de artiest R.E.M. voorkomen in de database.
71. Wat was de hoogste notering van een R.E.M.-nummer in de Top-2000 editie 2021?
    Gebruik **MIN** om de positie van het nummer te bepalen.
72. Maak een overzicht van artiesten en het aantal liedjes dat ze in de database hebben op volgorde van het aantal (aflopend). Welke artiest heeft de meeste titels? Hoeveel?
73. (EXTRA) Voorspel wat de volgende query doet en controleer daarna het resultaat:
    SELECT COUNT(DISTINCT artiest) FROM top2000

1.5 Een query in een query

Een fan van Di-rect vraagt zich af welk nummer van hen het laagst stond in de top 2000 van 2021. Ze voert de volgende query uit:

```
SELECT titel,ed2021 FROM top2000
WHERE artiest='Di-Rect' AND NOT ed2021 IS NULL
ORDER BY ed2021 DESC
```

titel	ed2021
Wild Hearts	1775
Devil Don't Care	1443
Times Are Changing	1067
Soldier On	14

FIGUUR 11

Het resultaat van deze query zie je in figuur 11. Deze query geeft vier resultaten. Stel nu dat we alleen de titel met de laagste notering (in de top 2000 is dat een hoog getal!) als resultaat willen.

Bij vraag 71 heb je de hoogste notering voor de band R.E.M. in 2021 gevonden. Dit was wel zonder titel:

```
SELECT MIN(ed2021) FROM top2000 WHERE artiest='R.E.M.'
```

Als we de titel van het laagst genoteerde nummer van Di-rect in 2021 willen vinden, ligt deze query misschien voor de hand:

```
-- LET OP de volgende query levert een foutmelding op!
SELECT MAX(ed2021),titel FROM top2000
WHERE artiest = 'Di-Rect'
ORDER BY ed2021 DESC
```

Error: In aggregated query without GROUP BY, expression #2 of SELECT list contains nonaggregated column 'top_2000_v1.top2000.titel'; this is incompatible with sql_mode=only_full_group_by

FIGUUR 12 FOUTMELDING!

Het resultaat in figuur 12 is teleurstellend: een foutmelding! Met SQL kun je wel de laagste positie vinden (als je het gedeelte *,titel* uit bovenstaande query verwijdert), maar om dit maximum te vinden moet hij wel alle records bekijken met Di-rect als artiest. Daarbij onthoudt hij alleen het maximum, maar niet de overige gegevens van het bijbehorende record.

De oplossing is om de zoekopdracht in stapjes op te delen: eerst zoeken we de laagste positie (1775) en daarna zoeken we in de database het liedje dat hierbij hoort. Je krijgt dan een query binnen een query:

```
SELECT titel,ed2021 AS laagst FROM top2000 WHERE ed2021 =
(
        SELECT MAX(ed2021) FROM top2000 WHERE artiest = 'Di-rect'
)
```

titel	laagst
Wild Hearts	1775

FIGUUR 13 HET JUISTE RESULTAAT

Let goed op de haakjes! Tussen de haakjes (vetgedrukt) wordt eerst opgezocht dat de laagste positie de waarde 1775 heeft. Deze waarde wordt in de eerste regel gebruikt voor *ed2021* om de titel erbij te zoeken.

Opdracht 13 Dubbele top 2000-query's

74. Maak een query die als enige resultaat de titel en positie geeft van de hoogste notering van de band R.E.M. (figuur 14) in de editie van de top 2000 van 2021.
75. Voorspel wat de volgende query doet en controleer daarna het resultaat:
    SELECT titel,ed2021 AS hoogst FROM top2000 WHERE artiest='R.E.M.' AND ed2021<
    (SELECT AVG(ed2021) FROM top2000
    WHERE artiest='R.E.M.')
76. (EXTRA) Selecteer alle titels van *The Beatles* die in 2020 hoger stonden dan het hoogst genoteerde nummer van R.E.M. van dat jaar. Om welke titels gaat het?

FIGUUR 14 R.E.M.

1.6 EXTRA Het gebruik van IN en HAVING

Een query levert een verzameling van nul of meer records of andere resultaten. Het is ook mogelijk om van twee query's de uitkomsten te vergelijken, om bijvoorbeeld te zien of er overlap in resultaten is of juist niet. We leggen dit uit aan de hand van de volgende vraag:
Welke artiesten stonden in 2021 in de top-100 van de top 2000, maar een jaar eerder nog niet?
(Nota bene: dat hoeft dus niet per se met dezelfde titel te zijn!)

We maken eerst een query die een lijst maakt van alle artiesten in de top-100 van 2020:

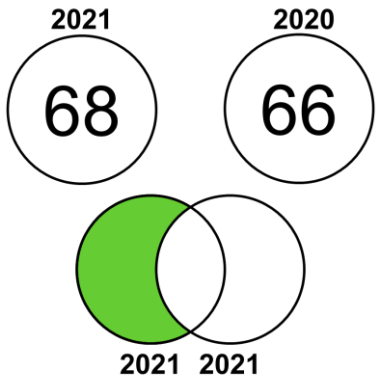
```
SELECT DISTINCT artiest FROM top2000 WHERE ed2020<=100          -- artiesten top 100 van 2020
```

Dit levert een lijst met 66 artiestennamen, omdat sommige artiesten met meerdere nummers zijn vertegenwoordigd. Van deze 66 namen moeten we uitzoeken of ze met een liedje in de top-100 van 2021 stonden. We zoeken de artiesten voor wie dit niet het geval is:

```
SELECT DISTINCT artiest FROM top2000 WHERE ed2021<=100          -- artiesten top 100 van 2021
AND artiest NOT IN                                              -- extra eis
(SELECT DISTINCT artiest FROM top2000 WHERE ed2020<=100)        -- artiesten top 100 van 2020
```

De derde regel bevat tussen haakjes de query waar we mee zijn begonnen: dit leverde 66 artiesten. De eerste regel selecteert artiesten die in 2021 een top-100-notering hadden (dat zijn 68 verschillende artiesten).

De tweede regel stelt de eis dat van deze tweede groep artiesten er niemand ook in de lijst artiesten van 2020 mag zitten. Figuur 15 markeert nu de artiesten waar we naar op zoek zijn. Het groene deel bevat de artiesten die wel in 2021 wel maar in 2020 niet in de top-100 stonden. De vraag is:
Wie van die 68 artiesten zit niet in de lijst met 66 artiesten?



FIGUUR 15

We gebruiken hiervoor **NOT IN**. Wil je juist op zoek naar artiesten die in beide resultaten terug komen, dan gebruik je **IN**.

Wanneer we een berekening uitvoeren met bijvoorbeeld **COUNT**, **AVG,MAX** of **MIN** en we willen dat de uitkomst een bepaalde eigenschap heeft, dan gebruiken we **HAVING**. Ook hier een voorbeeldvraag:
Welke artiesten staan met 20 titels of meer in de database?

Bij opdracht 12 hebben we een overzicht van het aantal titels per artiest gemaakt met de query:

```
SELECT artiest,COUNT(*) AS aantal FROM top2000 GROUP BY artiest ORDER BY aantal DESC
```

Hiermee krijgen we ook artiesten die maar met één of twee liedjes in de database staan. Met de toevoeging **HAVING** en de juiste eis kunnen we de voorbeeldvraag beantwoorden:

```
SELECT artiest,COUNT(*) AS aantal FROM top2000 GROUP BY artiest
HAVING aantal >= 20 ORDER BY aantal DESC
```

Kijk goed naar het vetgedrukte deel: de **alias** *aantal* die in de eerste regel wordt gemaakt, wordt in de tweede regel opnieuw gebruikt.

EXTRA opdracht 14: Top 2000 met IN en HAVING

77. Voer de query met de **NOT IN**-constructie uit de theorie uit. Hoeveel resultaten krijg je?
78. Harald zegt dat de query uit de vorige vraag hetzelfde resultaat oplevert als de query:
    SELECT DISTINCT artiest FROM top2000 WHERE ed2021<=100
    AND NOT ed2020<=100
    Vind jij dat ook? Zo niet, leg dan uit wat het verschil is in resultaat tussen de twee query's.
79. In de **NOT** IN-query van vraag 77 staat *artiest*. Vervang dit door *titel*.
    Verwacht je nu evenveel resultaten als bij de vorige vraag? Of meer of minder?
80. Genereer een overzicht van alle liedjes in de database van artiesten die in 2018 een top-10-notering hadden. Zorg voor een tabel met artiest, titel en het jaar (waarin het liedje gemaakt is: *uit*). Sorteer de tabel op naam van de artiest en zorg dat binnen die sortering de liedjes op volgorde van jaar staan.
81. Voer de query met de **HAVING**-constructie uit de theorie uit. Hoeveel resultaten krijg je?
82. Maak een overzicht van artiesten met meer dan drie titels in de top-100 van de top 2000 van 2018. Toon de naam van de artiest en het aantal titels, gesorteerd van hoog naar laag. Zijn er meer artiesten met hetzelfde aantal? Zorg dan dat ze in alfabetische volgorde worden getoond.
83. Hoeveel titels komen vaker dan één keer voor, denk je? Doe eerst een schatting en maak daarna een query die een overzicht geeft van de titel en het aantal keren dat hij in de database voorkomt. Uiteraard zet je de titel die het vaakst voorkomt bovenaan!

Extra opdracht 15: HAVING en ALL

Bij de vorige opdracht zochten we naar titels van liedjes waar meerdere artiesten mee in de top-2000 staan. Hoe selecteer je nu alleen die titel die het meest voorkomt?

Gebruik de database *top 2000 v1*.

84. Bekijk het resultaat van de query:
    SELECT titel FROM top2000 GROUP BY titel
    HAVING COUNT(titel) >= ALL
    (SELECT COUNT(titel) from top2000 GROUP BY titel)
    Wat is de betekenis van het vetgedrukte deel van bovenstaande query? Wat is het resultaat?
85. Maak een overzicht van de artiesten die een liedje hebben met de titel die het vaakst voorkomt (het resultaat van de vorige vraag) op volgorde van het jaar van uitgave.
    LET OP: je mag de titel niet in je query invullen. Het is de bedoeling dat je de query van de vorige vraag uitbreidt om tot je antwoord te komen.

FIGUUR 16

Extra opdracht 16: Rekenen met datums

Gebruik de database *weerstations*.

86. Voer de volgende query uit en leg aan de hand van de uitkomst uit wat de functie **DATEDIFF** doet:
    SELECT m_code, DATEDIFF(eind,start) as duur,eind,start FROM inzet WHERE eind>0
    ORDER BY duur DESC
87. Wat is de betekenis van het resultaat dat je nu hebt gekregen? En wat is daarbij de eenheid van de uitkomst van **DATEDIFF**?
88. Maak een overzicht van alle meters die meer dan duizend dagen hebben gewerkt, voordat ze werden vervangen.
89. Gebruik https://www.w3schools.com/sql/func_mysql_datediff.asp om een overzicht te genereren van meters die nog niet zijn vervangen en minder dan 500 dagen (sinds vandaag) in gebruik zijn, op volgorde van het aantal dagen (oplopend).

H2 MEERDERE TABELLEN

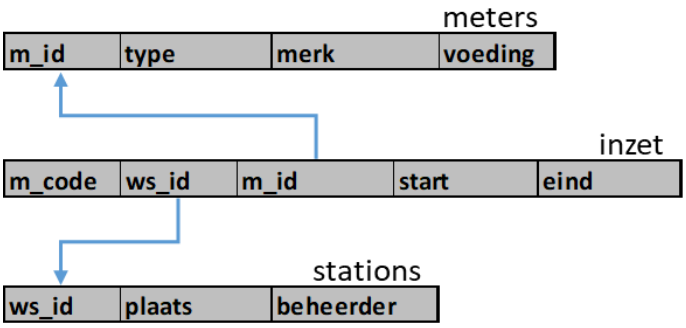
2.1 Relationele database: strokendiagram

Is het je opgevallen dat alle query's die je hebt gemaakt gebruik maken van één tabel? De database *top 2000 v1* heeft ook maar één tabel, maar *weerstations* bestaat uit drie tabellen (*stations*, *meters* en *inzet*). In de inleiding moest je gebruik maken van de relatie tussen die tabellen voor de vraag:

Geef het type en het merk van de meter die op dit moment het meest in gebruik is bij de weerstations.

Om deze opdracht uit te voeren moeten de tabellen *meters* en *inzet* worden gecombineerd. Dat kan alleen als je de relatie tussen de tabellen begrijpt.

De relatie tussen de tabellen uit de database *weerstations* staat in figuur 17 en heet een **strokendiagram**. De relatie tussen de tabellen geef je aan met een pijl die **verwijzing** of **referentie** heet.

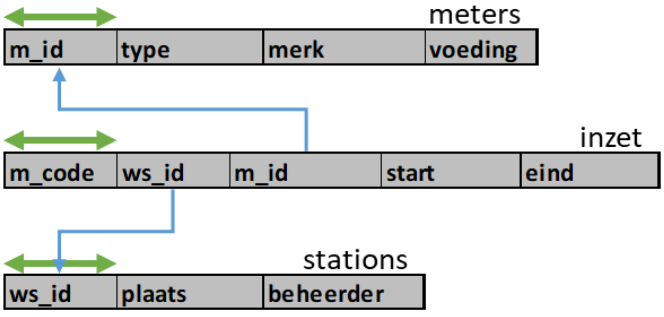


FIGUUR 17

Kijk goed naar de richting van de pijlen van de referenties. Waarschijnlijk had je zelf al gezien dat de waarden van *ws_id* in de tabel *inzet* terugverwijzen naar *ws_id* in de tabel *stations*. Op die manier kun je van een ingezette meter terugvinden bij welk weerstation hij wordt gebruikt. De pijl wijst daarom in de richting van de tabel *stations*. Daar is in eerste instantie de waarde van *ws_id* terug te vinden. De tabel *stations* is in dit geval een **parent**-tabel (Engels: ouder) en *inzet* is een **child**-tabel (Engels: kind).

Als een meter is ingezet, moet dat altijd bij één van de weerstations uit de database zijn. Anders gezegd: in de child-tabel *inzet* staan alleen waarden van *ws_id* die ook in de parent-tabel *stations* staan. Dit heet **referentiële integriteit**.

De *ws_id* is een code die hoort bij één **uniek** station. Als er meerdere stations met dezelfde *ws_id* zouden zijn, zou je nog steeds niet weten waar een meter is ingezet. Met die unieke code kun je het weerstation terugvinden. Het is de **primaire sleutel** voor de tabel *stations*. Merk op: in de tabel *inzet* is de *m_id* helemaal niet uniek: daar mag hij wel vaker voorkomen. In figuur 18 zijn voor alle tabellen de primaire sleutels toegevoegd aan het strokendiagram met een groene pijl.



FIGUUR 18

Het uniek zijn van een kolom of veld in een tabel heet **uniciteit**. Voor een sleutel geldt naast uniciteit ook de eis dat hij voor elk tabel-item een waarde moet hebben. Hij is niet *null*.

m_code	ws_id	m_id	type	merk	voeding	start	eind
3721	2	1	luchtdruk	Samsung	5V	2016-03-27	null
3412	4	2	luchtdruk	Samsung	12V	2005-10-04	2007-09-09
3681	3	2	luchtdruk	Samsung	12V	2005-10-04	2008-09-28
3924	5	2	luchtdruk	Samsung	12V	2005-10-04	2008-10-30
3512	4	4	luchtdruk	Philips	5V	2007-09-09	2011-12-20
3607	4	5	luchtdruk	Vavetech	5V	2011-12-20	null
3918	1	5	luchtdruk	Vavetech	5V	2007-07-14	null
3134	3	6	luchtdruk	Vavetech	12V	2008-09-28	null

FIGUUR 19

Waarom is er onderscheid gemaakt tussen de verschillende soorten meters die er bestaan en de inzet van deze meters in de weerstations? Figuur 19 laat zien hoe het eruit zou zien wanneer de tabellen *meters* en *inzet* zouden worden gecombineerd tot één tabel. Van de soort meter met *m_id* = 2 zijn er drie ingezet. Het oranje kader laat zien dat informatie over deze meter (zoals het merk) nu driedubbel in de database staat: er is overtollige informatie. Dit fenomeen heet **redundantie**. Dit kan leiden tot **inconsistentie**. Stel dat je bij één van deze records i.p.v. 12 V per ongeluk 13 V invoert. Dan krijgen we tegenstrijdige of inconsistente informatie, want twee dezelfde meters moeten logischerwijs ook werken op dezelfde spanning.

Opdracht 17: Kritisch kijken naar de top 2000 v1

Bekijk de structuur van de database *top 2000 v1*.

90. Is er in de database sprake van inconsistentie?
91. Is er in de database sprake van redundantie?
92. Is er een kolom in de tabel als primaire sleutel aan te wijzen? Zo ja, welke?
93. Bedenk een nadeel van de opbouw van deze tabel (deze database)?

Opdracht 18: De opbouw van top 2000 v2

Bekijk de structuur van de database *top 2000 v2*.

~~94. Er zijn twee tabellen met een kolom *jaar*. Wat is hun betekenis?~~
95. Teken (op papier of op de computer) het strokendiagram van de database *top 2000 v2*. Denk aan de referenties en sleutels.
96. Benoem voor alle tabellen of er een *parent* of *child* is en zo ja welke tabellen dat zijn.
97. Bestaat er voor deze database het gevaar van inconsistentie? Leg uit waarom wel of niet.
98. Noem argumenten waarom *top 2000 v2* beter is dan *top 2000 v1*.

Opdracht 19: Mogelijke fouten in top 2000 v2

Bij de volgende opdrachten worden situaties geschetst die zich in de database *top 2000 v2* zouden kunnen voordoen. Geef steeds aan of er in het voorbeeld sprake is van een (database-technisch) probleem is. Zo ja, leg dan uit of er sprake is van een probleem met referentiële integriteit, inconsistentie en / of redundantie.

FIGUUR 20 MOSS (NL)

99. De band *Foxygen* komt voor in de tabel *artiest* maar er bestaat geen liedje van hen in de tabel *titel*.
100. Er bestaat een liedje in de tabel *titel* zonder dat het bijbehorende artiest_*id* in de tabel *artiest* staat.
101. Er zijn twee bands met dezelfde naam *Moss* in de tabel *artiest*.
102. Er bestaat een liedje in de tabel *titel* zonder dat de bijbehorende *id* terug te vinden is in *notering*.
103. De tabel *notering* bevat twee records met verschillend *id* maar hetzelfde *lied_id*.
104. De tabel *notering* bevat twee records met verschillend *id* maar hetzelfde *lied_id* en *jaar*.

Opdracht 20: Uitbreiden van top 2000 v2

Bij de volgende opdrachten staan mogelijke uitbreidingen van de database *top 2000* v2. Ga uit van het strokendiagram dat je hierboven van *top 2000 v2* hebt gemaakt (of vraag de uitwerking) en pas het aan zodat aan de aanvullende eisen wordt voldaan.

FIGUUR 21 PAUL MCCARTNEY

105. We willen voor elke artiest aangeven wat het land van herkomst is.
106. We willen voor elk liedje aangeven wat de duur (in minuten en seconden) is.
107. Bij een artiestennaam horen soms meerdere bandleden. We willen voor elke artiestennaam de afzonderlijke voor- en achternamen en de geboortedatum van de meewerkende muzikanten kunnen opnemen in een tabel *personen*.
108. (EXTRA) Voor het geval je dat nog niet had geregeld bij de vorige vraag: sommige artiesten (zoals Paul McCartney van *The Beatles* en *Wings*) spelen in meerdere bands. Zorg voor een strokendiagram waarbij geen redundantie zal ontstaan.

2.2 Query's maken voor meerdere tabellen

Nu we iets weten over de opbouw van databases met meerdere tabellen, willen we er ook informatie uit kunnen halen. In deze paragraaf leer je de bijbehorende SQL-commando's.

Hiernaast zie je een eenvoudige database waarin van een aantal personen is geregistreerd welke hobby's ze hebben. Het veld *p_id* in de tabel *hobby* verwijst naar het veld *id* in de tabel *persoon*. Stel dat we met SQL willen vaststellen wat het antwoord is op de vraag:

Wat zijn de hobby's van Anja?

persoon	
id	**naam**
A	Anja
B	Bob
C	Cindy
D	Don

hobby	
p_id	**naam**
A	paard
A	tennis
B	gamen
B	voetbal
B	uitgaan
C	uitlapen
D	paard
D	uitgaan

FIGUUR 22 HOBBY'S

Als je in SQL wilt vertellen dat je meerdere tabellen wilt gebruiken, doe je dat simpelweg door ze op te sommen met een komma ertussen:

```
SELECT * FROM persoon,hobby
```

Het resultaat van deze query stelt wat teleur. De computer combineert alle records van de ene tabel met alle records van de andere tabel. Dit levert (4 · 8 =) 32 records op, waarvan er in figuur 23 een deel wordt getoond. Natuurlijk moeten we nog zeggen dat het om Anja gaat. Omdat er in beide tabellen een kolom met de titel *naam* is, zetten we de tabelnaam voor de kolomnaam met daartussen een punt:

```
SELECT * FROM persoon,hobby WHERE persoon.naam='Anja'
```

persoon,hobby			
id	**naam**	**p_id**	**naam**
A	Anja	A	paard
B	Bob	A	paard
C	Cindy	A	paard
D	Don	A	paard
A	Anja	A	tennis
B	Bob	A	tennis
C	Cindy	A	tennis
D	Don	A	tennis
A	Anja	B	gamen
B	Bob	B	gamen
C	Cindy	B	gamen
D	Don	B	gamen
A	Anja	B	voetbal

etcetera

FIGUUR 23

De computer zoekt nu alle resultaten uit figuur 23 waarvoor geldt dat de n*aam* Anja is. Zie figuur 24: het record met de *naam* Anja uit de tabel *persoon* wordt gecombineerd met alle records uit de tabel *hobby*.

Voor ons mensen is het duidelijk dat de *id* A van Anja in de tabel *persoon* hoort bij de *p_id* A in de tabel *hobby*, maar voor een computer niet. Daarom vertellen we met SQL dat ze gelijk moeten zijn:

```
SELECT * FROM persoon,hobby WHERE persoon.naam='Anja'
AND id=p_id
```

persoon,hobby			
id	**naam**	**p_id**	**naam**
A	Anja	A	paard
A	Anja	A	tennis
A	Anja	B	gamen
A	Anja	B	voetbal
A	Anja	B	uitgaan
A	Anja	C	uitlapen
A	Anja	D	paard
A	Anja	D	uitgaan

FIGUUR 24

Als deze query wordt uitgevoerd krijg je alleen nog de resultaten in het groene kader van figuur 24. Dit is de informatie waar we naar op zoek waren, maar de twee kolommen *id* en *p_id* staan wat slordig. In hoofdstuk 1 heb je geleerd dat je in plaats van de * ook kolomnamen kunt selecteren.

Omdat de kolom *naam* ook al in de tabel *persoon* staat, gebruiken we weer de combinatie van tabelnaam en kolomnaam met een punt:

```
SELECT hobby.naam FROM persoon,hobby WHERE persoon.naam='Anja' AND id=p_id
```

De resultaattabel bevat nu één kolom met de titel *naam* en twee records met 'paard' en 'tennis'. Vind je de titel *naam* verwarrend, dan zou je gebruik kunnen maken van een alias om de kolomnaam te veranderen:

```
SELECT hobby.naam AS hobby FROM persoon,hobby WHERE persoon.naam='Anja' AND id=p_id
```

Herinnering opdrachten met query's

Noteer alle query's en overige antwoorden in een schrift of in een Word-document! Wanneer gevraagd wordt om iets te selecteren, noteer je de query. Wordt er een andere vraag gesteld waarvoor je een query moet gebruiken om het antwoord te vinden, dan noteer je zowel de query als het antwoord.

Opdracht 21: Hobby's

Voor deze opgave gebruiken we de database uit de theorie (figuur 22).

109. De database bestaat uit twee tabellen met elk twee kolommen. Zijn er kolommen waarvoor uniciteit geldt? Zo ja, welke?
110. Zijn de combinaties van de kolommen uniek voor de tabel *persoon*? En de tabel *hobby*?

Bij de volgende opdrachten moet je een query schrijven met de voorkennis uit hoofdstuk 1. Omdat je de database niet digitaal hebt, kun je het niet uitproberen!

111. Selecteer de namen van de personen die uitgaan als hobby hebben.
112. Maak een alfabetisch overzicht van alle hobby's (waarin alle hobby's maar één keer voorkomen).
113. Als de vorige vraag, maar nu met een kolom waarin het aantal mensen staat dat de hobby heeft. Zorg dat de kolom ook de titel *aantal* krijgt.
114. Als de vorige vraag, maar toon nu alleen hobby's waarvoor het aantal groter dan 1 is.
    HINT: gebruik HAVING
115. (EXTRA) Selecteer alle namen die als derde letter een 'n' hebben.
116. (EXTRA) Omschrijf in algemene termen wat het resultaat zal zijn van de volgende query:
    SELECT naam FROM hobby GROUP BY naam HAVING COUNT(naam)>=ALL
    (SELECT COUNT(naam) FROM hobby GROUP BY naam)
117. (EXTRA) Omschrijf in algemene termen wat het resultaat zal zijn van de volgende query:
    SELECT DISTINCT persoon.naam FROM persoon,hobby WHERE id=p_id AND hobby.naam IN
    (SELECT naam FROM hobby GROUP BY naam HAVING COUNT(naam)>=ALL
    (SELECT COUNT(naam) FROM hobby GROUP BY naam))

Opdracht 22: Meerdere tabellen in weerstations

Gebruik de database *weerstations*.

118. Selecteer de metercodes (*m_code*) van meters die zijn ingezet (of geweest) bij het weerstation in Zuidhorn.
119. Selecteer de metercodes (*m_code*) en de naam van de beheerder van alle meters van weerstations in Groningen. Sorteer de lijst op de naam van de beheerder.
120. Maak een overzicht van namen van beheerders en het aantal meters dat ze hebben ingezet, op volgorde van aantal (aflopend).
121. Selecteer het merk, de voeding en de vervangingsdatum (*eind*) van alle luchtdrukmeters die vervangen zijn.
122. Maak een overzicht van alle merken en het aantal keren dat er een meter van dat merk vervangen is. Sorteer het overzicht op aantal, met de hoogste waarde bovenaan.
123. Welke meters zijn er op dit moment in gebruik (en dus nog niet vervangen) in Garrelsweer? Selecteer type, merk en voeding.
    Merk op: dit is de eerste query waar je drie tabellen voor nodig hebt!
124. Selecteer alle beheerders die een windkrachtmeter beheren.
125. (EXTRA) Selecteer alle beheerders die geen windkrachtmeter beheren.
126. (EXTRA) Geef type, merk en voeding van de meters die zijn vervangen. Maak ook een kolom duur die het aantal dagen geeft tot vervanging en sorteer de resultaten op deze duur (oplopend).

Extra opdracht 23: Complexe query's I

Gebruik de database *weerstations*.

Omschrijf in algemene termen wat het resultaat zal zijn van de volgende query's. Voer daarna de query uit en controleer of jouw omschrijving juist was.

127. SELECT * FROM meters WHERE m_id NOT IN
(SELECT m_id FROM inzet)
128. SELECT type,COUNT(type) AS aantal FROM meters WHERE m_id NOT IN
(SELECT m_id FROM inzet)
GROUP BY type ORDER BY aantal DESC
129. SELECT merk,COUNT(merk) AS aantal FROM meters WHERE m_id IN
(SELECT m_id FROM inzet)
GROUP BY merk HAVING aantal>3 ORDER BY aantal DESC
130. SELECT meters.m_id,type,merk,voeding,start,DATEDIFF(CURRENT_DATE,start) AS duur
FROM meters,inzet WHERE meters.m_id=inzet.m_id
AND eind IS NULL ORDER BY duur DESC
Probeer zelf te bedenken wat CURRENT_DATE betekent.
131. SELECT merk FROM meters WHERE m_id IN
(SELECT m_id FROM inzet WHERE m_code<=ALL
(SELECT m_code FROM inzet))

Opdracht 24: Meerdere tabellen in top 2000 v2

In hoofdstuk I heb je zoekopdrachten gemaakt op basis van de database *top 2000 v1*. Nu we een verbeterde versie (*v2*) van deze database hebben, zijn de query's bij die zoekopdrachten ook veranderd. In deze opgave beantwoord je een aantal zoekopdrachten uit hoofdstuk I opnieuw. Gebruik de database *top 2000 v2*.

132. Selecteer de titel van het nummer dat op positie 1323 stond in de top-2000 van het jaar 2013?
133. In welk jaar had Mr. Probz voor het eerst een notering in de top-2000?
Maak een overzicht van alle noteringen en sorteer deze op jaar (aflopend).
134. Hoeveel nummers van Adele die gemaakt zijn in 2011 hebben een notering gehad in de top-2000?
Selecteer alleen de titel.
135. (EXTRA) Welke artiesten hebben een notering gehad in de top-2000 met de titel *Home*, maar stonden niet in de editie van 2021? Deze query is lastiger in *v1*. Gebruik een query in een query.
136. Maak een overzicht van de top-2000 van 2020 met achtereenvolgens de kolommen *positie*, *artiest* en *titel*. Uiteraard zorg je dat de lijst gesorteerd is van laag naar hoog.
137. Welk nummer van John Legend stond in de editie van 2014 in de top-100 van de top-2000?
138. Selecteer alle nummers van Anouk die tussen 2005 en 2010 uitkwamen (2005 en 2010 doen mee!).
139. Welke artiesten hebben een liedje gemaakt dat na 2010 uit kwam en de titel *Sorry* had?
Maak een overzicht met alleen de naam van de artiest en het jaar waarin het liedje is gemaakt.
140. Maak een alfabetische lijst van alle artiesten die een notering in de top-20 van de top-2000 van 2014 hadden. Elke artiestennaam mag maar één keer voorkomen. Hoeveel artiestennamen vind je?
141. Maak een query die telt hoeveel liedjes van de artiest R.E.M. voorkomen in de database.
142. Wat was de hoogste notering van een R.E.M.-nummer in de Top-2000 van het jaar 2000? Maak een query die alleen dat hoogst genoteerde nummer als resultaat geeft.
143. Er staan 59 (!) liedjes van *The Beatles* in de database. Een aantal daarvan is in hetzelfde jaar gemaakt. Hoeveel verschillende jaren van uitgave zijn er voor liedjes van *The Beatles*?
Maak een aflopend gesorteerd overzicht van deze jaren van uitgave van liedjes van *The Beatles* met het aantal liedjes van dat jaar.
144. Maak een overzicht van artiesten en het aantal liedjes dat ze in de database hebben op volgorde van het aantal (aflopend). Welke artiest heeft de meeste noteringen? Hoeveel?
145. (EXTRA) Maak een overzicht van de artiesten die een liedje hebben met de titel die het vaakst voorkomt. HINT: gebruik HAVING

2.3 Normaliseren

In paragraaf 1 hebben we het strokendiagram van de database Weerstations bekeken (figuur 25). Waarom is de database op deze manier ontworpen?

In figuur 26 zie je de weerstations en de ingezette meters in één tabel. Er is sprake van redundantie: er zijn groepen waarin dezelfde gegevens zich herhalen. Dit is niet alleen overbodig, maar heeft ook het gevaar van inconsistentie in zich.

In figuur 26 zie je in de kolommen *plaats* en *beheerder* blokken die zich herhalen. In zo'n geval gaan we **afsplitsen**. Van deze kolommen maken we een nieuwe tabel *stations*. Omdat er geen kolom is die in zijn eentje uniciteit heeft, wordt vaak een identificatiecode (*id*; *identity*) als kolom toegevoegd aan de tabel die als sleutel fungeert. Zo'n kolom heet een **kunstmatige sleutel**.

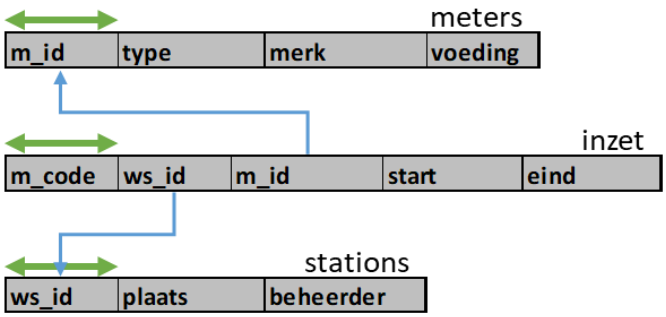
Als we de kolommen zomaar verwijderen, verliezen we informatie: we kunnen dan niet meer zien bij welk weerstation een meter is geplaatst. Daarom voegen we een kolom *ws_id* toe die verwijst naar de sleutel van de tabel *stations*.

Na de afsplitsing hebben we een kleinere tabel over die in figuur 27 met een andere sortering wordt getoond. De nieuwe kolom *ws_id* is hier extra gemarkeerd. Opnieuw zien we dubbelingen ontstaan (maar niet altijd!). Zo zijn er drie meters van het type met *m_id* = 2 ingezet, waarvan op dit moment drie keer wordt vermeld dat het een luchtdrukmeter van Samsung is die werkt op een voeding van 12 V.

Ook hier is het verstandig om kolommen (*m_id*, *type*, *merk* en *voeding*) af te splitsen naar een aparte tabel waarin de verschillende soorten *meters* als uniek record terugkomen.

In dit geval is er al een unieke identificatiecode *m_id*. Net als bij de stations moet er ook hier een kolom worden toegevoegd die de relatie tussen de *meters* en de *inzet* legt. Dit wordt de kolom *m_code*. Het is niet verplicht om een andere kolomnaam te kiezen; nog een keer *m_id* is ook toegestaan.

Met bovenstaande stappen hebben we de tabellen uit figuur 25 bereikt. Dit proces van ordenen en afsplitsen heet **normalisatie**. Bij grote hoeveelheden data met ingewikkelde koppelingen kan het een lastige klus zijn om te normaliseren. Hiervoor zijn aparte technieken en applicaties ontwikkeld die we hier niet zullen behandelen.



FIGUUR 25

WEERSTATIONS

m_id	type	merk	voeding	start	eind	plaats	beheerder
7	luchtdruk	Vavetech	5V	14-7-2007	*null*	Zuidhorn	Van der Veen
12	temperatuur	Vavetech	5V	14-7-2007	*null*	Zuidhorn	Van der Veen
5	luchtdruk	Vavetech	5V	14-7-2007	*null*	Zuidhorn	Van der Veen
15	temperatuur	Samsung	5V	27-3-2016	*null*	Groningen	Velthuizen
9	windkracht	Samsung	5V	27-3-2016	*null*	Groningen	Velthuizen
1	luchtdruk	Samsung	5V	27-3-2016	*null*	Groningen	Velthuizen
12	temperatuur	Vavetech	5V	7-1-2009	*null*	Groningen	Grgurina
6	luchtdruk	Vavetech	12V	28-9-2008	*null*	Groningen	Grgurina
8	windkracht	Vavetech	3V	4-10-2005	*null*	Groningen	Grgurina
2	luchtdruk	Samsung	12V	4-10-2005	28-9-2008	Groningen	Grgurina
13	temperatuur	Philips	5V	4-10-2005	7-1-2009	Groningen	Grgurina
2	luchtdruk	Samsung	12V	4-10-2005	9-9-2007	Bedum	Palsma
4	luchtdruk	Philips	5V	9-9-2007	20-12-2011	Bedum	Palsma
5	luchtdruk	Vavetech	5V	20-12-2011	*null*	Bedum	Palsma
15	temperatuur	Samsung	5V	1-1-2010	1-6-2010	Garrelsweer	Osinga
7	luchtdruk	Vavetech	5V	14-7-2017	*null*	Garrelsweer	Osinga
8	windkracht	Vavetech	3V	1-1-2010	*null*	Garrelsweer	Osinga
12	temperatuur	Vavetech	5V	4-10-2005	*null*	Garrelsweer	Osinga
2	luchtdruk	Samsung	12V	4-10-2005	30-10-2008	Garrelsweer	Osinga

FIGUUR 26

INZET METERS

m_id	ws_id	type	merk	voeding	start	eind
1	2	luchtdruk	Samsung	5V	27-3-2016	*null*
2	3	luchtdruk	Samsung	12V	4-10-2005	28-9-2008
2	4	luchtdruk	Samsung	12V	4-10-2005	9-9-2007
2	5	luchtdruk	Samsung	12V	4-10-2005	30-10-2008
4	4	luchtdruk	Philips	5V	9-9-2007	20-12-2011
5	1	luchtdruk	Vavetech	5V	14-7-2007	*null*
5	4	luchtdruk	Vavetech	5V	20-12-2011	*null*
6	3	luchtdruk	Vavetech	12V	28-9-2008	*null*
7	1	luchtdruk	Vavetech	5V	14-7-2007	*null*
7	5	luchtdruk	Vavetech	5V	14-7-2017	*null*
8	3	windkracht	Vavetech	3V	4-10-2005	*null*
8	5	windkracht	Vavetech	3V	1-1-2010	*null*
9	2	windkracht	Samsung	5V	27-3-2016	*null*
12	1	temperatuur	Vavetech	5V	14-7-2007	*null*
12	3	temperatuur	Vavetech	5V	7-1-2009	*null*
12	5	temperatuur	Vavetech	5V	4-10-2005	*null*
13	3	temperatuur	Philips	5V	4-10-2005	7-1-2009
15	2	temperatuur	Samsung	5V	27-3-2016	*null*
15	5	temperatuur	Samsung	5V	1-1-2010	1-6-2010

FIGUUR 27

Als je scherp hebt gekeken, heb je opgemerkt dat de informatie in figuur 26 nog niet eens volledig is. In de database weerstations staan in de tabel *meters* namelijk ook meters die wel op de markt zijn maar niet zijn ingezet bij één van de stations. Hoe zou jij die informatie in de tabel van figuur 26 hebben verwerkt?

Opdracht 25: Mediatheek

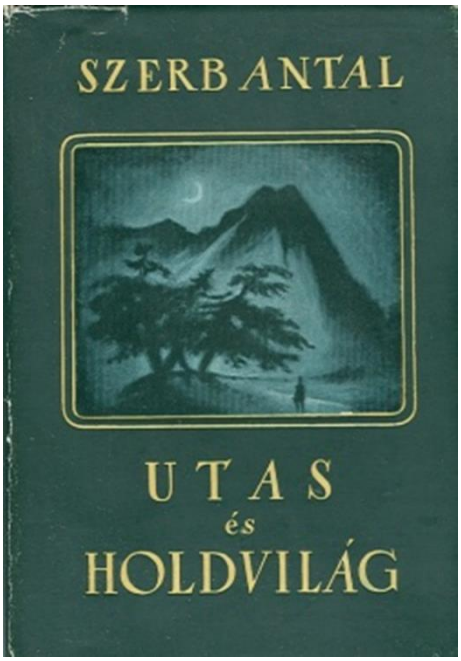
Een mediatheek gebruikt een Excel-bestand om bij te houden welke klanten welke boeken lenen. Hierbij wordt geregistreerd op welke datum een boek wordt geleend. Als het boek terug is wordt een einddatum genoteerd. In figuur 28 zie je een klein deel van de tabel uit het Excel-bestand.

MEDIATHEEK							
naam	plaats	leeftijd	boektitel	uitgave	auteur	geleend	terug
René	Zuidhorn	16	Het leven dat wij droomden	1931	Roelants	22-3-2017	29-3-2017
Jelma	Groningen	16	Het leven dat wij droomden	1931	Roelants	5-4-2017	
Wibe	Drachten	15	Kind onder Kannibalen	1982	Bukowski	26-4-2017	6-5-2017
René	Zuidhorn	16	Komen en gaan	1964	Roelants	5-4-2017	
Joke	Zoutkamp	19	Postkantoor	1971	Bukowski	25-1-2017	3-2-2017
Kees	Bedum	15	Postkantoor	1971	Bukowski	12-4-2017	26-4-2017
Kees	Bedum	15	Postkantoor	1971	Bukowski	26-4-2017	6-5-2017
Joke	Zoutkamp	19	Publieke Werken	1999	Rosenboom	13-1-2017	27-1-2017
Joke	Zoutkamp	19	Publieke Werken	1999	Rosenboom	27-1-2017	4-2-2017
Jelma	Groningen	16	Reis bij maanlicht	1937	Szerb	17-2-2017	25-2-2017

FIGUUR 28

De mediatheek wil overstappen op een database.

146. Bestudeer de tabel en ontwerp op basis van de gegevens een database. Kijk goed welke kolommen afgesplitst kunnen worden naar een nieuwe tabel. Geef deze tabellen een naam en teken vervolgens het bijbehorende strokendiagram, inclusief uniciteit en referenties.



FIGUUR 29

Nu de medewerkers van de mediatheek het resultaat van de vorige vraag zien, willen ze nog meer data gaan registreren. Hieronder staan een aantal nieuwe wensen. Geef van elk van de drie wensen aan of hij zal zorgen voor een extra tabel in de database en zo ja **waarom**.

147. In plaats van de leeftijd van een klant wil de mediatheek de geboortedatum gaan bewaren. Daarnaast moet de volledige naam worden geregistreerd alsmede de initialen.
148. Van de schrijvers wil men ook de voornaam en het land van herkomst opnemen.
149. Er werken drie mediathecarissen in de mediatheek, waarvan voor-, achternaam en telefoonnummer worden opgeslagen. Bij het uitlenen van een boek wil men in de toekomst bijhouden welke mediathecaris dit heeft afgehandeld.
150. (EXTRA) Breid jouw strokendiagram uit zodat aan de aanvullende wensen wordt voldaan.

Opdracht 26: Weerstations

Gebruik de database *weerstations*.

In de theorie wordt benadrukt dat er metertypes in de tabel *meters* staan die bij geen enkel weerstation zijn ingezet.

151. Met welke query kunnen de bijbehorende *m_id*'s worden gevonden?
152. Stel dat we de contactgegevens van de producenten (zoals *Samsung*) van meters (naam bedrijf, e-mail) in de database willen registreren. Beschrijf hoe dan het strokendiagram van de database *weerstations* uit de theorie moet worden aangepast.

2.4 Zelf databases maken en aanpassen

In de vorige paragraaf ben je bezig geweest met databasedesign: het correct ontwerpen van gegevensbanken. Met SQL kun je een strokendiagram omzetten naar een echte database. Ook is het mogelijk om de database te vullen met records of gegevens te wijzigen of te verwijderen. Organisaties zijn zuinig op hun data. Het kan erg gevaarlijk of kostbaar zijn om een record, een tabel of zelfs een hele database weg te gooien. Daarom worden de **rechten** van een gebruiker vaak beperkt. Dit betekent dat het voor een gebruiker verboden is om bepaalde commando's uit te voeren. Wie alleen **leesrecht** heeft kan alleen query's uitvoeren om gegevens op te vragen. Met **schrijfrecht** kun je ook wijzigingen aanbrengen.

Voor de volledigheid laten we hier zien hoe je met SQL een database kunt maken en kunt vullen met tabellen en records. In de praktijk wordt dit vaak met een speciaal programma gedaan i.p.v. met SQL. Verderop in dit hoofdstuk leer je hoe je een database maakt met de online *tool* **MyAdmin**.

Een database maken of verwijderen doe je zo:

```
CREATE DATABASE school                    -- Maak een nieuwe database met de naam school
DROP DATABASE school                      -- Verwijder de database met de naam school
```

De database bevat nu nog geen tabellen. Een nieuwe tabel voeg je toe op de volgende manier:

```
CREATE TABLE leerlingen (                 -- Maak een nieuwe tabel met de naam leerlingen
  leerlingnummer int(6) NOT NULL AUTO_INCREMENT,   -- Maak een kolom leerlingnummer
  voornaam varchar(32) NOT NULL,
  tussenvoegsel varchar(16),
  achternaam varchar(32) NOT NULL,
  geboortedatum date NOT NULL,
  PRIMARY KEY (leerlingnummer)            -- De kolom leerlingnummer is de primaire sleutel
 )

DROP TABLE leerlingen                     -- Verwijder de tabel met de naam leerlingen
```

In dit voorbeeld staan veel begrippen die we nog niet eerder hebben gezien. Bij het maken van de tabel worden kolommen aangemaakt. Na de titel van de kolom (b.v. *leerlingnummer*) volgt het type data met tussen haakjes de maximale grootte van de informatie. Met *int(6)* wordt een kolom gemaakt om een *integer* (getal) in op te slaan dat maximaal uit 6 cijfers mag bestaan. Met *varchar(32)* maak je een kolom waarin ook andere tekens mogen voorkomen en met *date* rekent de database op invoer in de vorm van een datum. Notabene: er zijn veel meer dataformaten mogelijk dan we hier laten zien. Lees daarvoor LINK.

Met de toevoeging **NOT NULL** geef je aan dat je eist dat er voor elk record in de tabel een waarde moet zijn. In het voorbeeld zie je dat iedereen een *voornaam* en *achternaam* moet hebben, maar niet iedereen heeft een *tussenvoegsel* (b.v. Willem *van* Oranje) in zijn naam. Veel tabellen bevatten een kunstmatige sleutel in de vorm van een identificatiecode (*id*). Dan kan **auto_increment** (automatisch ophogen) worden toegevoegd. In ons geval zal de database automatisch het *leerlingnummer* met 1 ophogen als een nieuw record (dus een nieuwe leerling) aan de database wordt toegevoegd.

De tabel bevat nu nog geen data. Een record aanmaken doe je met:

```
INSERT INTO leerlingen (leerlingnummer,voornaam,tussenvoegsel,achternaam,geboortedatum)
VALUES (148438,'Dafne',NULL,'Schippers','1992-06-15')

INSERT INTO leerlingen (voornaam,tussenvoegsel,achternaam,geboortedatum)
VALUES ('Willem','van','Oranje','1533-04-24')
```

Merk op dat bij de tweede opdracht helemaal geen leerlingnummer wordt meegegeven. Door de **AUTO_INCREMENT** zal *Willem van Oranje* automatisch *leerlingnummer 148439* krijgen.

Opdracht 27: Records wijzigen en verwijderen

Selecteer in de *querier* de (nu nog) lege database *Project*.

153. Maak een tabel *leerlingen* door de code uit de theorie te kopiëren en daarna uit voeren.
154. Selecteer in de *querier* opnieuw de database *Project* en stel vast of de tabel *leerlingen* is toegevoegd aan de database.
155. Voeg jezelf toe als leerling aan de database. Noteer de bijbehorende SQL-opdracht.
156. Voeg *Willem van Oranje* aan de database toe met de SQL-opdracht uit de theorie.
157. Voer de query
SELECT * FROM LEERLINGEN
uit en bepaal wat het leerlingnummer van de andere leerling is.

FIGUUR 30 WILLEM VAN ORANJE

In de theorie staan SQL-opdrachten om records toe te voegen. Maar wat nu als je een foutje hebt gemaakt of een leerling uit de lijst moet worden verwijderd?

158. Voer de volgende SQL-opdracht uit:
UPDATE leerlingen **set** tussenvoegsel='de', achternaam='Zwijger' **WHERE** achternaam='Oranje'
159. Geef jezelf in de database een nieuwe voornaam. Noteer de bijbehorende SQL-opdracht.
160. Voer de volgende SQL-opdracht uit:
DELETE FROM leerlingen **WHERE** voornaam='Willem'
161. Verwijder jezelf uit de tabel *leerlingen*. Noteer de bijbehorende SQL-opdracht.
162. Verwijder de tabel *leerlingen*. Noteer de bijbehorende SQL-opdracht. Stel vast dat *Project* nu een lege database is.

Opdracht 28: Zelf een database vullen

In §2.2 hebben we gewerkt met de twee tabellen in figuur 31. De tabellen hebben een relatie. We gaan de database *Project* vullen met deze tabellen.

persoon

id	naam
A	Anja
B	Bob
C	Cindy
D	Don

hobby

p_id	naam
A	paard
A	tennis
B	gamen
B	voetbal
B	uitgaan
C	uitlapen
D	paard
D	uitgaan

FIGUUR 31 HOBBY'S

163. Teken het bijbehorende strokendiagram. Vergeet niet om voor elke tabel de uniciteit te markeren.
164. Maak de tabel *hobby* in de database *Project*. Noteer de bijbehorende SQL-opdracht.

Als je records wilt toevoegen aan de database, hoef je dat niet één voor één te doen. Bekijk de volgende SQL-opdracht maar eens:

```
INSERT INTO hobby (p_id,naam)
VALUES ('A','paard'),('A','tennis'),('B','gamen'),('B','voetbal'),
('B','uitgaan'),('C','uitslapen'),('D','paard'),('D','uitgaan')
```

165. Voer bovenstaande query uit en controleer of de records zijn toegevoegd aan de tabel *hobby*.
166. Voeg de tabel *persoon* toe. Noteer de bijbehorende SQL-opdracht.
167. Voeg de records uit de tabel *persoon* toe met één opdracht. Noteer de bijbehorende SQL-opdracht.
168. Selecteer alle hobby's van Bob.
169. Selecteer de *naam* van alle personen die als hobby *paard* hebben.
170. Anja en Don stoppen met hun hobby *paard*. Geef de SQL-opdracht die de bijbehorende records uit de database verwijdert. Je mag gebruik maken van het feit dat er nu niemand meer is die *paard* als hobby heeft.
171. (EXTRA) Bekijk de LINK en probeer een aantal SQL-opdrachten uit voor de database *Projects*.
172. Open *leesmij.html* en klik op *Databases opnieuw aanmaken*. Kies de eerstgenoemde actie.

2.5 Computerpracticum: MyAdmin

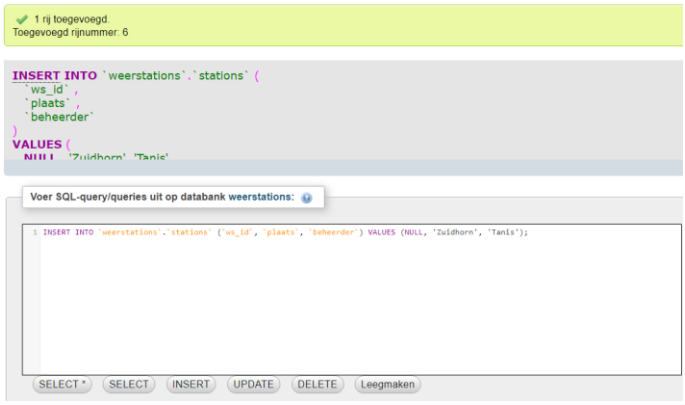
De tool **MyAdmin** is op veel webservers geïnstalleerd om databases te maken en te beheren. Tijdens dit computerpracticum leer je alleen de basisbeginselen.

i) Open *PHPMyAdmin* via de startpagina van onze werkomgeving en log in. Zie eventueel LEESMIJ.txt voor instructies om in te loggen.
ii) Klik in het linker deelvenster op *weerstations*.

We zijn nu aanbeland in het onderdeel *structuur* van MyAdmin. Op dit moment toont de pagina de structuur van de database *weerstations*. Deze bevat de drie tabellen *inzet*, *meters* en *stations*.

iii) Klik achter *stations* op *Verkennen*. Je ziet nu de inhoud van de tabel *stations*.
iv) Klik bovenaan op het tabblad *Structuur*. Je ziet nu de structuur van de tabel *stations*.
v) Klik bovenaan op het tabblad *Invoegen* en voeg jezelf toe als beheerder van een weerstation in je eigen woonplaats. LET OP: je hoeft geen waarde voor *ws_id* in te vullen, want de kolom *ws_id* heeft *AUTO_INCREMENT* als instelling, waardoor je automatisch een eigen *ws_id* krijgt toegewezen.

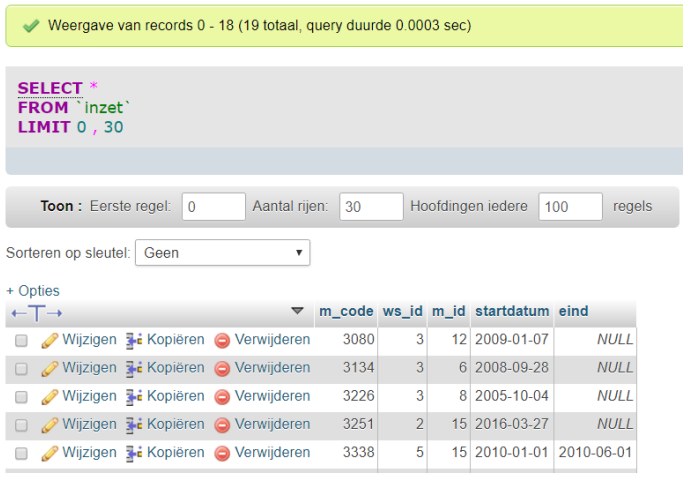
Als je bij de vorige opdracht op *START* hebt geklikt, dan zie je een scherm zoals deels hiernaast staat afgebeeld. MyAdmin heeft jouw handeling zelf omgezet in een uitgebreide SQL-opdracht. Als je bovenaan kijkt, zie je ook dat nu het tabblad *SQL* is geselecteerd.



FIGUUR 32

vi) Voer in dit scherm een query uit die de naam van alle beheerders in jouw woonplaats selecteert.
vii) Selecteer de tabel *inzet* en klik op het tabblad *structuur*.
viii) Klik achter *start* op *Veranderen* en verander de naam van *startdatum*.
ix) Verander de naam van *eind* in *einddatum* en verken daarna de tabel *inzet*.

Het scherm waar je nu bent is deels weergegeven in de figuur hiernaast. Je kunt hier individuele records manipuleren.



FIGUUR 33

x) Wat gebeurt er als je op een kolomnaam klikt?
xi) De meter met de laagste *m_code* is kapot gegaan. Wijzig de einddatum naar de datum van vandaag.
xii) Gebruik het tabblad *Zoeken* om alle records in de tabel *inzet* te vinden met *ws_id* = 4.
xiii) Klik links op *weerstations* om opnieuw de tabellenstructuur van de hele database te selecteren.
xiv) Gebruik in het hoofdscherm onderaan de optie *Tabel aanmaken*. Maak een nieuwe tabel *merk* met 4 kolommen.
xv) Probeer zelfstandig de kolommen *merk_id* (primaire sleutel & AUTO_INCREMENT), *merknaam*, *telefoon* en *e-mail* aan te maken. Bekijk hierbij de keuzes die je bij *Type* kunt maken.
xvi) (EXTRA) Pas de tabel *meters* aan, zodanig dat deze een relatie krijgt met de nieuwe tabel *merk*. Vul hiertoe de tabel *merk* met de juiste records.

Als je klaar bent, willen we de database *weerstations* graag weer in zijn oorspronkelijke staat herstellen.

xvii) Ga naar de startpagina van onze omgeving en klik op *Herstel en opslag databases*. Kies vervolgens voor *opnieuw aanmaken van de door de docent aangeleverde databases*.

ANTWOORDEN

Antwoorden Inleiding

1. Data zijn gegevens. Bij informatie wordt er betekenis aan die gegevens gekoppeld. Als mens kun je de data dan begrijpen, zodat het iets oplevert: informatie.
2. Het volgende niveau is *wijsheid*. Volgens sommige wetenschappers zit hier nog de stap *begrip* tussen. De volgorde wordt dan: 1) data, 2) informatie, 3) kennis, 4) begrip & 5) wijsheid.
3. Jouw surfgedrag op het internet levert veel data op die wordt gebruikt om reclame aan te bieden die zo goed mogelijk op jou is afgestemd.
   De staafjes en kegeltjes in je oog (netvlies) leveren grote hoeveelheden data die door een netwerk van zenuwen in je oog en de hersenen wordt omgezet naar informatie.
4. Bij *Big data* gaat het om erg grote hoeveelheden data die vaak snel en in grote hoeveelheden tegelijk worden verzameld. Ze kunnen alleen met digitale systemen worden geanalyseerd.
5. WhatsApp, Albert Heijn, Google, etc. etc. Jij kunt er waarschijnlijk zo meer dan tien noemen!
6. Albert Heijn gebruikt jouw aankoopgegevens voor reclame op maat, voorraadbeheer, aanbiedingen. Een concurrent van Albert Heijn zou dezelfde gegevens kunnen gebruiken om meer klanten te trekken.
7. –
8. Dat zijn er twee: van het type luchtdruk en type temperatuur zijn er allebei zeven varianten.
9. Er zijn in totaal zes vervangingen geweest. Dat zie je aan de kolom *eind*. Vier daarvan waren luchtdrukmeters.
10. Het type meter met *m_id* 2 en 12 komen allebei drie keer voor in de tabel inzet. Maar de meters met *m_id* 2 zijn allemaal al vervangen, dus het gaat om *m_id* 12. Dit is een temperatuurmeter van het merk Vavetech.
11. Voor het eerst vervangen kijken we naar de kolom *eind*. Op 9 september 2007 was de eerste vervanging met *m_id* 12. Dit is een temperatuurmeter van het merk Vavetech.
    Voor de snelste vervanging moeten we kijken naar het verschil tussen *begin* en *eind*. Dit verschil is het kleinst voor een meter met *m_id* 15. Dit is een temperatuurmeter van het merk Samsung.
12. Er zijn drie meters met 12V-voeding met *m_id* 2, 6 en 16. Meter 2 is niet meer in gebruik, meter 16 is nooit gebruikt, dus het antwoord is slecht één meter (namelijk meter 6).
13. De meters met *m_id* 8, 11, 17 & 18 werken op 3V. Meter 8 wordt gebruikt door de weerstations met *ws_id* 3 en 5. Meter 11 en 18 zijn nooit gebruikt. Meter 17 wordt ook door het station met *ws_id* 5 gebruikt. Bij deze weerstations horen de beheerders Grgurina (3) en Osinga (5).
14. –

Antwoorden H1

15. –
16. SELECT * FROM meters WHERE voeding=’12V’
17. SELECT m_id FROM meters WHERE type='temperatuur' AND merk='Samsung' AND voeding='5V'
18. SELECT m_code,start,eind FROM inzet WHERE m_id=15
    Dit levert twee records:
    3251 met start op 27-3-2016 en nog geen einddatum
    3338 met start op 1-1-2010 en eind 1-6-2010
19. SELECT * from meters where type='luchtdruk' AND merk='Vavetech' AND voeding='5V'
20. –
21. SELECT * FROM top2000 WHERE ed2021=1627
    Antwoord: *In your eyes* van *The Weeknd*
22. SELECT * FROM top2000 WHERE artiest=’Maan’
    Het antwoord op de vraag is: 2020
23. SELECT titel, uit FROM top2000 WHERE artiest='Adele'
    Dit levert vier records.

24. SELECT * FROM top2000 WHERE titel='Sorry'
    Antwoord: Justin Bieber
25. –
26. Q1: alle meters die werken op 3V of 12V
    Q2: alle weerstations die niet in de plaats Groningen gevestigd zijn
    Q3: alle ingezette meters met een metercode die kleiner is dan 3200 (3200 doet niet mee)
    Q4: alle ingezette meters die nog niet zijn vervangen (omdat het veld eind niet is gevuld).
    Q5: alle weerstations met een ws_id die groter of gelijk is aan 4 (4 doet wel mee)
    Q6: alle meters, behalve de meters met een voeding van 5V die niet van Samsung zijn
27. –
28. SELECT * FROM meters WHERE type='windkracht' AND m_id<10
29. SELECT * FROM meters WHERE type='temperatuur' AND NOT merk='Samsung'
30. A: alle luchtdrukmeters en alle meters van Philips, behalve luchtdrukmeters van Philips
    B: alle meters van Philips, behalve die van het type luchtdrukmeter
    C: (trap er niet in) dit zijn gewoon alle luchtdrukmeters
31. A: SELECT * FROM meters WHERE (type='luchtdruk' OR merk='Philips')
       AND NOT (type='luchtdruk' AND merk='Philips')
    B: SELECT * FROM meters WHERE merk='Philips' AND NOT type='luchtdruk'
    C: SELECT * FROM meters WHERE type='luchtdruk'
32. –
33. Alle meters, behalve de luchtdrukmeters van Philips
34. Alle meters die niet van het type luchtdruk of van Philips zijn
35. Hetzelfde als bij de vorige vraag! Dus: Alle meters die niet van het type luchtdruk of van Philips zijn
36. SELECT ed2017, artiest, titel FROM top2000 WHERE NOT ed2017 IS null
37. SELECT ed2017, titel, artiest FROM top2000 WHERE NOT ed2017 IS null AND ed2017 <= 1000 AND artiest = 'Oasis'
    Het antwoord is: *Wonderwall*
38. SELECT uit, titel, artiest FROM top2000 WHERE artiest = 'Anouk' AND uit > 2009
    Twee records: *Birds* en *For bitter or worse*
39. SELECT * FROM top2000 WHERE artiest='Anouk' AND uit>=2005 AND uit<=2010
    Je hoort zeven records te vinden, inclusief drie nummers uit het jaar 2005 en één uit 2010
40. SELECT * FROM top2000 WHERE artiest='Stromae' OR artiest='Maroon 5'
41. SELECT artiest, uit FROM top2000 WHERE titel = 'one' AND uit < 2000
    Dit levert twee records: Metallica & U2
42. SELECT artiest, titel, uit, ed2017, ed2018 FROM top2000 WHERE uit=1976 AND ed2017<=1500 AND (ed2018 > 1500 OR ed2018 IS null)
    Dit levert vijf records.
43. SELECT beheerder FROM stations ORDER BY beheerder
44. SELECT * FROM inzet ORDER BY start
45. Behalve op plaats, is de lijst nu ook nog op beheerder gesorteerd. Het effect is dat de twee records behorende bij stations in Groningen nu van volgorde zijn verwisselt.
    Door aan ORDER BY een tweede kolom toe te voegen (met een komma) kun je dus binnen een sortering (op plaats) nog een tweede sortering (op beheerder) toevoegen.
46. A: De query sorteert de meters alfabetisch op type en daarbinnen op merk.
    B: De query sorteert de meters alfabetisch op merk en daarbinnen op type.
47. Er worden nog maar drie kolommen getoond in de volgorde merk, type, voeding
    De merken worden omgekeerd alfabetisch (aflopend) weergegeven. Binnen die sortering wordt het type meter alfabetisch (oplopend) weergegeven en daarbinnen de voeding weer omgekeerd alfabetisch.
48. De computer heeft alfabetisch aflopend gesorteerd en niet qua getalswaarde. Bij temperatuurmeters van Samsung zie je het goed: eerst 5V, dan 3V en dan pas 12V.
49. SELECT DISTINCT type FROM meters
50. De query geeft een overzicht van unieke combinaties van meters met een bepaalde voeding, gesorteerd op het type meter.
51. SELECT titel,jaar FROM top2000 WHERE artiest='Michael Jackson' ORDER BY jaar DESC
52. SELECT artiest,titel FROM top2000 WHERE artiest='The Police' OR artiest='Sting' ORDER BY titel

53. SELECT artiest,titel,jaar FROM top2000
    WHERE artiest='The Police' OR artiest='Sting' ORDER BY uit
    Ja, dat lukt ook als je de kolom *uit* niet opneemt in de SELECT.
54. SELECT DISTINCT artiest FROM top2000 WHERE ed2018<=20 order by artiest
    Dit levert 16 records of unieke artiesten.
55. SELECT DISTINCT uit FROM top2000 WHERE artiest='Racoon' order by uit DESC
    De tien liedjes zijn uitgekomen in zeven verschillende jaren.
56. Met het %-teken geef je aan dat op die plaats in de zoekterm nul, één of meerdere willekeurige tekens mogen staan.
57. SELECT DISTINCT artiest FROM top2000 WHERE artiest LIKE '%i'
    Er zijn 13 unieke artiestennamen die eindigen op i.
58. Alle artiestennamen die beginnen met The en eindigen op Brothers met daartussen een willekeurige hoeveelheid willekeurige tekens. (Dat zijn vijf artiesten.)
59. Met de _ geef je aan dat op die plaats precies één willekeurige teken moet staan.
60. SELECT artiest,titel FROM top2000 WHERE titel LIKE '%christmas%'
    Dit levert zeven records.
61. Bij query B eis je een nog een spatie achter *The*.
62. SELECT artiest,titel FROM top2000 WHERE artiest LIKE 'The%' AND NOT artiest like 'The %'
    Er zijn twee artiestennamen die je wel met query A vindt, maar niet met query B: *Them* en *Therapy?*
63. SELECT DISTINCT artiest FROM top2000 WHERE artiest LIKE '___'
    Dit levert acht verschillende artiestennamen van drie letters.
64. SELECT artiest,titel FROM top2000 WHERE artiest LIKE '%q%' AND titel LIKE '%q%'
    Dit levert twee records voor *q* (en maarliefst 151 voor de *y*).
65. –
66. SELECT COUNT(*) FROM meters WHERE voeding='3V'
67. SELECT merk,COUNT(*) AS aantal FROM meters GROUP BY merk ORDER BY aantal DESC
68. De query zoekt de meter uit met de hoogste m_code en geeft behalve die code ook de startdatum.
69. SELECT m_id,COUNT(*) AS N FROM inzet GROUP BY m_id ORDER BY N DESC
    Uit het resultaat is te zien de meters met *m_id* 2 en 12 beide drie keer zijn ingezet.
70. SELECT COUNT(*) FROM top2000 WHERE artiest='R.E.M.' (Dit levert negen titels).
71. SELECT MIN(ed2021) FROM top2000 WHERE artiest='R.E.M.'
    De hoogste positie van R.E.M. in 2021 was 103.
72. SELECT artiest,COUNT(*) AS aantal FROM top2000 GROUP BY artiest ORDER BY aantal DESC
    The Beatles hebben de meeste noteringen met liefst 40 verschillende liedjes.
73. De query telt het aantal unieke (verschillende) artiesten in de database.
74. SELECT titel,ed2021 AS hoogst FROM top2000 WHERE ed2021 =
    (SELECT MIN(ed2021) FROM top2000 WHERE artiest = 'R.E.M.')
    Het nummer *losing my Religion* stond in 2021 als hoogst genoteerde R.E.M.-nummer op 103.
75. Dit levert alle noteringen van R.E.M. in 2021 die beter waren dan de gemiddelde notering van alle nummers van R.E.M. in dat jaar.
76. SELECT titel,ed2020 FROM top2000 WHERE artiest='The Beatles'
    AND ed2020 < (SELECT MIN(ed2020) FROM top2000 WHERE artiest='R.E.M.')
    Het gaat om de nummers *Hey Jude*, *Let it be* en *Here comes the sun*.
77. De query geeft vijf namen van artiesten die wel in 2021 maar niet in 2020 in de top 100 stonden. (Eric Clapton, Procol Harum, Volbeat, Floor Jansen & Henk Poort en Nothing but Thieves)
78. Bij de query van Harald wordt per record gekeken of aan alle eisen is voldaan. Het kan best zo zijn dat een artiest zowel in 2021 als in 2020 in de top-100 stond, maar met een andere titel. De query van Harald telt dan toch een resultaat voor het liedje dat niet meer in de top-100 staat, waardoor deze query (mogelijk) een hoger aantal records oplevert (in dit geval 7 t.o.v. 5).
79. We gebruiken de query:
    SELECT DISTINCT titel FROM top2000 WHERE ed2021<=100
    AND titel NOT IN
    (SELECT DISTINCT titel FROM top2000 WHERE ed2020<=100)
    Dit levert acht namen van liedjes die wel in 2021 maar niet in 2020 in de top 100 van de top 2000 stonden. Dat is dus meer dan toen we de artiesten in de lijst vergeleken.
80. SELECT artiest, titel, uit FROM top2000 WHERE artiest IN (

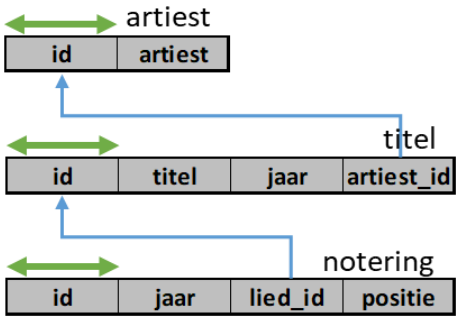
SELECT DISTINCT(artiest) FROM top2000 WHERE ed2018 <= 10) ORDER BY artiest, uit
De query levert 141 titels!

81. De query levert negen artiesten op met 20 of meer titels in de database.
82. SELECT artiest,COUNT(*) AS aantal FROM top2000 WHERE ed2019 <= 100
    GROUP BY artiest HAVING aantal > 2 ORDER BY aantal DESC, artiest
    Dit levert elf artiestennamen
83. SELECT titel,COUNT(*) AS aantal FROM top2000
    GROUP BY titel HAVING aantal > 1 ORDER BY aantal DESC
    Je krijgt nu 74 verschillende titels die meer dan één keer in de top-2000-database staan! *Crazy*…
84. De query tussen haakjes (derde regel) maakt een lijst van titels met het aantal keren dat deze titel voorkomt. Merk op dat in de eerste regel eigenlijk dezelfde query staat! Maar er is wel een toevoeging gedaan in de tweede regel. Het vetgedrukte deel eist dat het aantal keren dat het liedje voorkomt groter of gelijk moet zijn dan alle andere gevonden aantallen. Dat is natuurlijk alleen waar wanneer je het grootste aantal hebt gevonden. Daarom worden alleen titels getoond met dit grootste aantal.
85. SELECT artiest, uit FROM top2000 WHERE titel IN (
    SELECT titel AS aantal FROM top2000 GROUP BY titel
    HAVING COUNT(titel) >= ALL
    (SELECT COUNT(titel) from top2000 GROUP BY titel)
) ORDER BY uit
    Je vindt een lijst met twaalf artiestennamen.
86. DATEDIFF berekent het verschil tussen twee datums
87. De query geeft een overzicht van alle meters die zijn vervangen, op volgorde van de duur die ze zijn ingezet, waarbij de meter die het langst heeft gewerkt tot zijn vervanging bovenaan staat.
    De eenheid van DATEDIFF is het aantal dagen.
88. SELECT m_code, DATEDIFF(eind,start) as duur,eind,start FROM inzet WHERE eind > 0 HAVING duur > 1000 ORDER BY duur DESC
    Dit levert vier meters met *m_code* 3512, 3847, 3924 & 3681
89. SELECT m_code, DATEDIFF("2017-07-21",start) as duur,eind,start FROM inzet
    WHERE eind IS NULL HAVING duur < 500 ORDER BY duur
    Dit levert vier meters met *m_code* 3664, 3251, 3356 & 3721

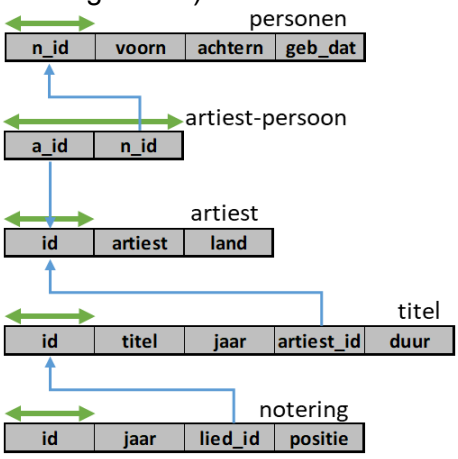
Antwoorden H2

90. Voor ons mensen is er wel een relatie tussen de records (b.v. verschillende liedjes van dezelfde artiest), maar voor de database niet. Dat betekent dat er ook geen inconsistentie kan zijn, want daarvoor met er gekoppelde informatie zijn die gelijk zou moeten zijn, maar waar dat niet zo is.
91. Als mens zien wij dat bijvoorbeeld bandnamen vaker worden genoemd, maar de informatie is daarmee nog niet redundant (overbodig), want voor elk liedje heb je die informatie wel nodig. Het zou wel efficiënter kunnen door de informatie te splitsen, waarbij je een aparte tabel maakt voor de artiesten. Je kunt dan met een nummertje verwijzen naar een artiest. Dit voorkomt dat elke keer opnieuw de naam van de artiest moet worden ingevoerd.
92. Een primaire sleutel moet uniek zijn. Er is geen kolom met uniciteit. De combinatie van *titel* en *artiest* zal bijna altijd uniek zijn, maar zelfs daar moet je oppassen.
    (Zo'n gecombineerde sleutel wordt in de praktijk wel eens gebruikt, maar staat niet in de theorie vermeld.)
93. Er staan allemaal nullen in de tabel, als een liedje in een bepaald jaar niet in de top-2000 stond. Het is erg lastig om uit te zoeken hoe vaak een liedje in de top-2000 heeft gestaan, wat de gemiddelde notering of hoogste notering was.
94. In de tabel *notering* slaat *jaar* op het jaar van uitzending van de top 2000.
    In de tabel *titel* slaat *jaar* op het jaar waarin het liedje is gemaakt.

~~95. Zie de tekening hiernaast.~~



96. De tabel *artiest* heeft *titel* als child (en *notering* als grand child).
    De tabel *titel* heeft *artiest* als parent en *notering* als child.
    De tabel *notering* heeft *titel* als child (en *artiest* als grandchild).
97. In theorie is het mogelijk om in de tabel *notering* twee records (met verschillend *id*) te maken met hetzelfde *lied_id* en *jaar*, maar verschillende positie. In werkelijkheid zou dit inconsistent zijn, want één liedje staat nooit op twee verschillende posities in de top-2000 van één specifiek jaar. Maar voor deze database is het strikt genomen niet inconsistent, omdat we niet de eis hebben gesteld dat de combinatie van *lied_id* en *jaar* uniek moet zijn.
    (Dat had wel gekund, maar zo'n gecombineerde sleutel hebben we niet geleerd.)
98. De bij de vorige opdracht opgesomde nadelen heb je hier niet.
99. Dat ligt niet voor de hand, maar is geen probleem met de referentiële integriteit, want er zijn geen records in de database met een verwijzing naar een record dat niet bestaat.
100. Dit is een probleem met referentiële integriteit: er is een verwijzing naar iets dat niet bestaat.
101. Dit komt in het echt vaker voor en is voor de database geen probleem, zolang we in de tabel *artiest* voor elke artiest maar een uniek *id* aanmaken. Dat laatste eisen we, omdat *id* de primaire sleutel van de tabel is.
102. Dat ligt niet voor de hand, maar is geen probleem met de referentiële integriteit, want er zijn geen records in de database met een verwijzing naar een record dat niet bestaat.
103. Dit zal zelfs vaak voorkomen! Dit betekent alleen maar dat een liedje meerdere jaren in de top-2000 heeft gestaan.
104. (zie hierboven) In theorie is het mogelijk om in de tabel *notering* twee records (met verschillend *id*) te maken met hetzelfde *track_id* en *editie*, maar verschillende positie. In werkelijkheid zou dit inconsistent zijn, want één liedje staat nooit op twee verschillende posities in de top-2000 van één specifiek jaar. Maar voor deze database is het strikt genomen niet inconsistent, omdat we niet de eis hebben gesteld dat de combinatie van *lied_id* en *jaar* uniek moet zijn.
    (Dat had wel gekund, maar zo'n gecombineerde sleutel hebben we niet geleerd.)
105. Zie de tekening hiernaast. Hierin is een veld *land* toegevoegd aan de tabel *artiest*.
106. De *duur* is toegevoegd aan de tabel *titel*.
107. Dit is de oplossing die ook voor vraag 107 geldt. Een eenvoudigere variant zou zijn om de tabel *artiest-persoon* weg te laten en in plaats daarvan de tabel *personen* uit te breiden met een veld *a_id*.
108. Nadeel van de bij de vorige vraag gesuggereerde eenvoudigere variant is dat je voor een artiest die in meerdere bands voor twee verschillende waarden van *a_id* dezelfde persoonsgegevens moet invullen. Dat is redundantie met het gevaar op inconsistentie. De oplossing hiernaast heeft dat gevaar niet.



109. Alleen de kolom *id* moet uniek zijn. Voor *persoon.naam* geldt dit niet, omdat er meerdere mensen met dezelfde naam kunnen zijn. Dat de kolommen in de tabel *hobby* niet uniek zijn, zie je al aan de records.
110. Omdat de kolom *id* al uniek is, is de combinatie van kolommen in de tabel *persoon* vanzelfsprekend ook uniek. Het zou ook logisch zijn dat de combinatie van kolommen in de tabel *hobby* uniek zijn, omdat je geen dubbele vermelding wil van het feit dat 'Anja als hobby tennis heeft.' Of de maker van de database dit ook heeft afgedwongen kunnen we aan de figuur niet zien.
111. SELECT persoon.naam FROM persoon,hobby WHERE id=p_id AND hobby.naam='uitgaan'
112. SELECT DISTINCT naam FROM hobby ORDER BY naam
113. SELECT naam,COUNT(*) AS aantal FROM hobby GROUP BY naam ORDER BY naam
114. SELECT naam,COUNT(*) AS aantal FROM hobby GROUP BY naam HAVING aantal>1
115. SELECT naam FROM persoon WHERE naam LIKE '__n%'
116. De query geeft een lijst hobby's die het populairst zijn (het meest worden gedaan)
117. De query geeft een lijst van namen van mensen die hobby's uitoefenen die het populairst zijn.

118. SELECT m_code FROM stations,inzet WHERE stations.ws_id=inzet.ws_id AND plaats='Zuidhorn'
     Dit levert drie meters met *m_code* 3582, 3637 & 3918.
119. SELECT m_code,beheerder FROM stations,inzet
     WHERE stations.ws_id=inzet.ws_id AND plaats='Groningen' ORDER BY beheerder
     Dit levert een lijst met acht meters van twee beheerders.
120. SELECT beheerder,COUNT(inzet.ws_id) AS aantal FROM stations,inzet
     WHERE stations.ws_id=inzet.ws_id GROUP BY beheerder ORDER BY aantal DESC
     Osinga 5, Grgurina 5, Van der Veen 3, Velthuizen 3, Palsma 3.
121. SELECT merk,voeding,eind FROM meters,inzet
     WHERE eind IS NOT null AND type='luchtdruk' AND meters.m_id=inzet.m_id
     Dit levert een lijst met vier meters (waarvan drie dezelfde Samsung 12V-meters).
122. SELECT merk,COUNT(merk) AS aantal FROM meters,inzet
     WHERE eind IS NOT null AND meters.m_id=inzet.m_id
     GROUP BY merk ORDER BY aantal DESC
     Samsung 4, Philips 2 (Opmerking: er is nog geen meter van Vavetech vervangen, maar dat zie je op deze manier niet terug!)
123. SELECT type,merk,voeding FROM stations,meters,inzet
     WHERE stations.ws_id=inzet.ws_id AND meters.m_id=inzet.m_id
     AND plaats='Garrelsweer' AND eind IS NULL
     Je krijgt een overzicht met drie verschillende Vavetech-meters.
124. SELECT beheerder FROM stations,inzet,meters
     WHERE stations.ws_id=inzet.ws_id AND meters.m_id=inzet.m_id
     AND type='windkracht'
     Dit zijn Velthuizen, Grgurina & Osinga
125. SELECT beheerder FROM stations WHERE beheerder NOT IN
     (SELECT beheerder FROM stations,inzet,meters
     WHERE stations.ws_id=inzet.ws_id AND meters.m_id=inzet.m_id
     AND type='windkracht')
     Dit zijn Palsma & Van der Veen
126. SELECT type,merk,voeding,DATEDIFF(eind,start) AS duur FROM meters,inzet
     WHERE meters.m_id=inzet.m_id AND eind IS NOT NULL ORDER BY duur ASC
127. Dit geeft een overzicht van meters die nog nooit zijn ingezet.
128. Dit geef het aantal niet ingezette meters van een bepaald type met het type met de meeste niet ingezette meters bovenaan.
129. Dit geeft de merken waarvan meer dan drie soorten (niet het aantal!) meters daadwerkelijk zijn ingezet (geweest).
130. De query geeft een alle gegevens van meters die op dit moment in bedrijf zijn met het aantal dagen dat ze al in bedrijf zijn, aflopend gesorteerd op dit aantal.
131. De query selecteert het merk van de meter met de laagste *m_code*.
132. SELECT titel,uit
     FROM notering, track
     WHERE track.track_id = notering.track_id
     AND editie = 2013
     AND positie = 1323
     Antwoord: *You'll never walk alone*
133. SELECT titel, positie, editie
     FROM artiest, notering, track
     WHERE artiest.artiest_id = track.artiest_id
     AND track.track_id = notering.track_id
     AND naam = 'Mr. Probz'
     ORDER BY editie ASCSELECT titel, positie, editie
     FROM artiest, notering, track
     WHERE artiest.artiest_id = track.artiest_id
     AND track.track_id = notering.track_id
     AND naam = 'Mr. Probz'

ORDER BY editie ASC
Antwoord: 2013

134. SELECT DISTINCT titel
FROM artiest, track
WHERE artiest.artiest_id = track.artiest_id
AND naam = 'Adele'
AND track.uit = 2011
Antwoord: 4 titels.
135. SELECT naam
FROM artiest, track
WHERE artiest.artiest_id = track.artiest_id
AND titel = 'home'
AND track_id NOT IN (
SELECT track_id
FROM notering
WHERE editie = 2021
)
Dit levert twee record. Simply Red en Edward Sharpe & The Magnetic Zeros
136. SELECT positie, naam AS artiest, titel
FROM artiest, notering, track
WHERE artiest.artiest_id = track.artiest_id
AND track.track_id = notering.track_id
AND editie = 2020
ORDER BY positie ASC
137. SELECT
138. SELECT titel, positie
FROM artiest, notering, track
WHERE artiest.artiest_id = track.artiest_id
AND track.track_id = notering.track_id
AND naam = 'John Legend'
AND editie = 2014
AND positie <= 100
Het antwoord is: *All Of Me* (op positie 74)
139. SELECT naam, uit
FROM artiest, track
WHERE artiest.artiest_id = track.artiest_id
AND titel = 'Sorry'
AND uit > 2010
Antwoord: Justin Bieber, Kensington en Nothing but Thieves
140. SELECT DISTINCT(naam)
FROM artiest, track, notering
WHERE artiest.artiest_id = track.artiest_id
AND track.track_id = notering.track_id
AND editie = 2014
AND positie <= 20
Je vindt 16 verschillende artiesten
141. SELECT COUNT(naam) AS aantal
FROM artiest, track
WHERE artiest.artiest_id = track.artiest_id
AND naam = 'R.E.M.'
Het aantal is 9
142. SELECT titel,positie,editie
FROM artiest, track, notering
WHERE artiest.artiest_id = track.artiest_id
AND track.track_id = notering.track_id
AND naam = 'R.E.M.'

```
AND editie = 2000
AND positie = (
SELECT MIN(positie)
FROM artiest, track, notering
WHERE artiest.artiest_id = track.artiest_id
AND track.track_id = notering.track_id
AND naam = 'R.E.M.'
AND editie = 2000
)
```

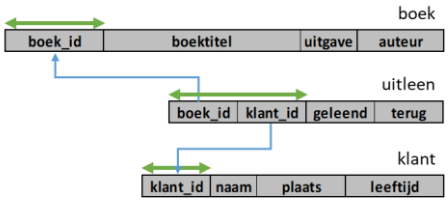
143.
```
SELECT uit, COUNT(uit) AS aantal
FROM artiest, track
WHERE artiest.artiest_id = track.artiest_id
AND naam = 'The Beatles'
GROUP BY uit
ORDER BY uit DESC
```

144.
```
SELECT COUNT(naam) AS aantal, naam
FROM artiest, track
WHERE artiest.artiest_id = track.artiest_id
GROUP BY naam
ORDER BY aantal DESC
```

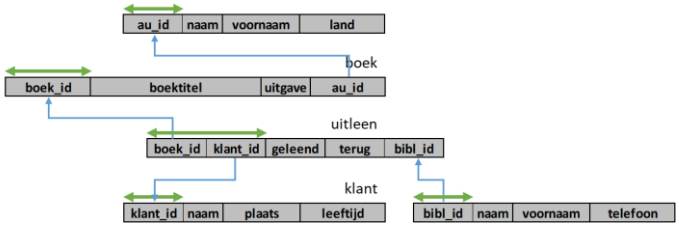
145.
```
SELECT naam
FROM artiest,track
WHERE artiest.artiest_id = track.artiest_id
AND titel IN (
SELECT titel
FROM track
GROUP BY titel
HAVING COUNT(titel) >= ALL (
SELECT COUNT(titel) AS aantal
FROM track
GROUP BY titel
)
)
```

Je vindt: Patsy Cline, Icehouse, Seal, Aerosmith en Gnarls Barkley

146. Zie de afbeelding.



147. Afsplitsen is niet nodig. De gegevens kunnen aan de tabel *klant* worden toegevoegd.
148. Afsplitsen naar een tabel *auteurs* is nodig, omdat er auteurs soms meerdere boeken schrijven. Dit zou in de tabel *boek* leiden tot redundantie.
149. Zou alleen de naam van de bibliothecaris worden geregistreerd, dan zou nog kunnen worden volstaan met een toevoeging in de tabel *uitleen*, maar aangezien er meerdere gegevens per bibliothecaris moeten worden vastgelegd en elke bibliothecaris meer dan één boek zal uitlenen is afsplitsen wenselijk om redundantie te voorkomen.
150. Zie de tekening hiernaast.
Merk op dat in de tekst alleen stond dat bijgehouden moet worden welke bibliothecaris een boek uitleent. In deze database kan niet worden bijgehouden welke bibliothecaris het boek weer in ontvangst neemt als het wordt teruggebracht!

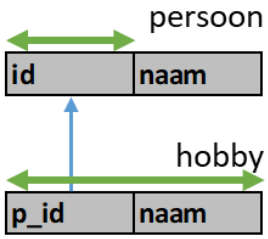


151.
```
SELECT m_id FROM meters
WHERE m_id NOT IN (SELECT m_id FROM inzet)
```

152. De kolom *merk* uit de tabel *meters* kan worden vervangen door een uniek *merk_id* die moet gaan verwijzen naar een *merk_id* in een tabel *merken*. Naast deze kolom moet deze nieuwe tabel

bestaan uit de kolommen *merknaam*, *telefoon* en *e-mail*. De kolom *merk_id* is de (kunstmatige) primaire sleutel van de tabel *merken*.

153. –
154. –
155. –
156. –
157. –
158. –
159. –
160. –
161. –
162. –
163. Zie de tekening hiernaast. De kolom *id* is de primaire sleutel in de tabel *persoon*. Merk op dat de hieraan gekoppelde *p_id* in de tabel *hobby* niet uniek is. Immers: iemand kan best twee hobby's hebben. Alleen de combinatie van de persoon en hobby (*p_id* & *naam*) is uniek.



164. CREATE TABLE hobby (
p_id varchar(2) NOT NULL,
naam varchar(32) NOT NULL
)
165. –
166. CREATE TABLE persoon (
id varchar(2) NOT NULL,
naam varchar(32) NOT NULL,
PRIMARY KEY (id)
)
167. INSERT INTO persoon (id,naam)
VALUES ('A', 'Anja'), ('B', 'Bob'),('C', 'Cindy'), ('D', 'Don')
168. SELECT hobby.naam FROM persoon,hobby WHERE id=p_id AND persoon.naam='Bob'
169. SELECT persoon.naam FROM persoon,hobby WHERE id=p_id AND hobby.naam='paard'
170. DELETE FROM hobby WHERE naam='paard'