van data naar info

Domein H: databases

INHOUD

H1 datawereld 3
1.1 Inleiding 3
1.2 Data bewaren 4
1.3 Databasemodellen 7

**“A database administrator walks into a NoSQL bar,
but he turn leaves because he can’t find a table.”**

— ERLEND OFTEDAL

**“Smart data structures and dumb code work
a lot better than the other way around.”**
— ERIC S. RAYMOND

**“About 15% of all Google search queries are queries that no one
has ever searched before. That means every day, humans around the world
are googling millions of unprecedented queries.”**
— QUINCY LARSON

“Don’t ask SQL developers to help you move furniture. They drop tables.”
— CARLA NOTAROBOT (TWITTER ACCOUNT)

H1 DATAWERELD

1.1 Inleiding

In 2019 meldde Google dat het 1,1 miljard euro wilde investeren in nieuwe datacenters in Nederland zoals in de Eemshaven (Groningen; zie figuur 1.1). Hiermee wordt de totale investering van alleen Google al twee-en-half miljard! Wat wordt er in deze datacenters bewaard? En hoe wordt het bewaard? En waarom heeft het zoveel waarde?

FIGUUR 1.1

Dit zijn vraagstukken die we in deze module gaan behandelen, beginnend met de basisvraag: wat is **data**? Data zijn *ruwe* gegevens, zoals die bijvoorbeeld rechtstreeks uit een temperatuursensor komen. Een verzameling van data of gegevens wordt ook wel een **dataset** genoemd.

In figuur 1.2A zie je een dataset. Het probleem met deze data is, dat de gegevens onbruikbaar zijn, omdat ze niet interpreteerbaar zijn: je weet niet wat ze voorstellen. Gaat het hier b.v. om plekken op aarde? Er is geen context of betekenis, waardoor de data geen waarde heeft.

Peru	205	133	63
Sienna	160	82	45
Tan	210	180	140

A

kleur	R	G	B
Peru	205	133	63
Sienna	160	82	45
Tan	210	180	140

B

FIGUUR 1.2

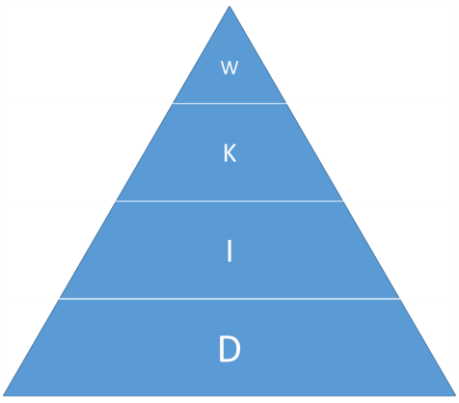
Om de data betekenis te geven, is meer data nodig. In figuur 1.2B is betekenis aan de originele ruwe data gegeven door **metadata** toe te voegen: extra data om andere data te beschrijven. De gegevens zijn nu herkenbaar geworden: er is sprake van **informatie**. Het zijn kleurnamen met hun RGB-kleurcode (rood, groen en blauw).

Google levert informatie via de bekende zoekmachine op basis van heel veel data. Met informatie kun je nader onderzoek doen, in de vorm van analyses. Door verschillende informatieonderdelen met elkaar te verbinden of patronen te ontdekken, ontstaat een nieuw begrip en inzicht in een onderwerp: er ontstaat **kennis**. Kennis is het begrijpen van de informatie. Overigens bestaan er meerdere definities van kennis. Soms wordt aan de reeks data, informatie en kennis nog **wijsheid** toegevoegd. Informatie is beschrijvend (Engels: *what is?*), kennis is vooral gericht op de informatie die we al hebben (verklarend: Engels: *why is?*) en wijsheid is gericht op verwachtingen en nieuwe keuzes (op basis van ervaring en patronen: *why do?*).

Opdracht 1 DIKW-piramide

Met betrekking tot de theorie, spreekt men over de **DIKW-piramide**.

1. Waar staan de letters van de afkorting DIKW voor?
2. Op het internet kun je plaatjes vinden van de DIKW-piramide, vergelijkbaar met figuur 1.3. Leg uit waarom hiervoor een piramidevorm wordt gebruikt.
3. Wat is het verschil tussen data en informatie?
4. Noem vijf bedrijven die data van jou in hun bezit hebben.
5. Op basis van vraag 4: welke informatie kunnen die bedrijven hieruit afleiden (Denk ook aan combinaties van data)?
6. Stel dat jij een ruim budget had om een grote dataset te kopen. Welke data zou je willen hebben? En met welk doel?



FIGUUR 1.3

Opdracht 2 Wat doen we met al die data?

Bekijk de video waarin, op licht-filosofische wijze, wordt gesproken over het gebruik van data, interpretatie en de wijsheid die daar uit zou kunnen volgen. De lezing bevat de nodige *oneliners*.

7. Noteer de drie uitspraken of zinnen die je het meest interessant of treffend vindt.

1.2 Data bewaren

Deze paragraaf gaat niet over het bewaren van data op een harde schijf of SD-kaart. Het gaat hier niet om de techniek van de fysieke opslag, maar om welke indeling of **datastructuur** je kunt gebruiken om vergelijkbare data altijd op dezelfde manier op te slaan. De centrale vraag daarbij is: *Wat wil je met de opgeslagen informatie doen?* Want bewaren is meestal geen doel op zich en soms verboden.

kleur	R	G	B
Peru	205	133	63
Sienna	160	82	45
Tan	210	180	140

FIGUUR 1.4

Er bestaat niet één datastructuur voor de opslag van data. Daarom kijken we in dit hoofdstuk naar meerdere aanpakken (of **paradigma**'s). De keuze voor een paradigma hangt in de praktijk af van vragen als:

- Hoeveel data is er of wordt er verwacht?
- Hoe vaak verandert de dataset? (b.v. veel lezen of schrijven?)
- Voor wie moet de data beschikbaar zijn?
- Welk doel heeft de opslag? Wat wordt er straks mee gedaan? (Wordt er bijvoorbeeld straks statistiek mee bedreven?)

kleur	Peru
R	205
G	133
B	63

kleur	Sienna
R	160
G	82
B	45

kleur	Tan
R	210
G	180
B	140

FIGUUR 1.5

In figuur 1.4 zie je nogmaals de data uit figuur 1.2. De data wordt weergegeven in een **tabel**. Het betreft een tabel met drie kleuren, waarbij aangegeven is wat de kleurnaam (*kleur*) is en met hoeveel rood, groen en blauw (de primaire keuren uit hun RGB-code) de kleur is opgebouwd. Rijen in dit soort tabellen, worden **records** genoemd.

In figuur 1.5 staat dezelfde data niet in een tabel, maar in de vorm van objecten. De term **object** ken je misschien al van object-georiënteerd programmeren. Vaak wordt één unieke eigenschap van het object gebruikt om het object te identificeren. In dit geval is dit de eigenschap of het **attribuut** *kleur*.

Bij de programmeerlessen heb je vast ook al kennis gemaakt met de **variabele** en de **array** of lijst. Het uitvoeren van programmeercode met daarin een variabele, array (lijst) of object zorgt voor de opslag van data in het werkgeheugen van de computer. Als je een hoeveelheid data langer wil bewaren in een extern geheugen zoals een harde schijf, dan gebruik je daarvoor een **bestand** of een **database**. Deze module gaat alleen over databases.

Databases zijn er in verschillende soorten, afhankelijk van de gekozen opslagstructuur ofwel het gekozen **databasemodel** (zie § 1.3). Databases die werken met tabellen zoals in figuur 1.4 heten **relationele databases**, omdat elk record bestaat uit datapunten die een **relatie** met elkaar hebben (bij elkaar horen).

Hoofdstuk 2 is volledig gewijd aan relationele databases. Daarnaast bestaan er databases waarin data op een andere manier wordt opgeslagen, zoals met de objecten in figuur 1.5. Deze niet-relationele databases heten in het algemeen **noSQL**-databases.

Opdracht 3 data in een object

In figuur 1.6 zie je nogmaals de data uit figuur 1.5. Aan de data is de bijbehorende hexadecimale kleurcode toegevoegd, die je misschien kent van het maken van websites. Deze hex-code kun je zelf afleiden (als je weet hoe) uit de gegeven hoeveelheden rood, groen en blauw.

Peru	
R	205
G	133
B	63
hex	CD853F

Sienna	
R	160
G	82
B	45
hex	A0522D

Tan	
R	210
G	180
B	140
hex	D2B48C

FIGUUR 1.6

8. Noem een argument om de hex-code niet in een database op te slaan en een argument om dit wel te doen.
9. In figuur 1.6 is behalve de hex-code nog meer informatie toegevoegd. Welke informatie is dat?
10. Welke metadata zie je in figuur 1.6?
11. Welke metadata zie je wel in figuur 1.4 maar niet in figuur 1.6?
12. Gebruik het internet om informatie te vinden over de kleur *Gainsboro*. Maak hiermee zowel een object als een record.

Opdracht 4 XML I: een dataset beschrijven

In figuur 1.6 is de data (en de metadata) gerepresenteerd als een plaatje (met tekst) van drie losse objecten. Een manier om data *echt* op te slaan, is door het als platte tekst te bewaren met een vaste structuur. Dit kan worden gedaan met behulp van **XML**.

In figuur 1.7 zie je een voorbeeld van een bestand in XML-formaat. Het zijn gegevens van twee dieren uit de praktijk van een dierenarts.

13. Waar staat XML voor?
14. Hoe kun je zien dat het om twee dieren gaat?
15. Op welke manier is ervoor gezorgd dat duidelijk is welke data bij welk dier hoort?
16. XML wordt zelfbeschrijvend (Engels: *self-descriptive*) genoemd. Wat wordt daarmee bedoeld?

```
<praktijk>
    <dier>
        <naam>Buck</naam>
        <klant>Al Bundy</klant>
        <soort>hond</soort>
        <geslacht>m</geslacht>
        <geboren>1983</geboren>
    </dier>
    <dier>
        <naam>Illbatting</naam>
        <klant>Pippi</klant>
        <soort>paard</soort>
        <geslacht>m</geslacht>
        <geboren>1961</geboren>
    </dier>
</praktijk>
```

FIGUUR 1.7

Opdracht 5 XML II: de structuur van XML

In deze opgave gaan we een XML-bestand bestuderen en aanpassen.

17. Open het XML-bestand *H1O05_dierenarts_1.0.xml* in je browser. Jouw browser geeft de data weer met behulp van de symbolen ► en ▼. Klik op deze symbolen: wat zie je?

Data wordt in XML beschreven met **elementen** zoals `<naam>Buck</naam>` die zijn gemaakt met een *start-tag* (`<naam>`), een *eind-tag* (`</naam>`) en de data (**Buck**).

Boven de data staat in de meeste browsers (zie figuur 1.8):

This XML file does not appear to have any style information associated with it. The document tree is shown below.

18. Wat wordt bedoeld met deze boomstructuur (Engels: *tree*)?
19. Met betrekking tot de structuur van data, kom je vaak de Engelse termen *parent* en *child* tegen. Wat wordt daarmee bedoeld? Gebruik eventueel het internet.
20. Noem een element dat zowel een *parent* als een *child* is.

This XML file does not appear to have any style information associated with it. The document tree is shown below.

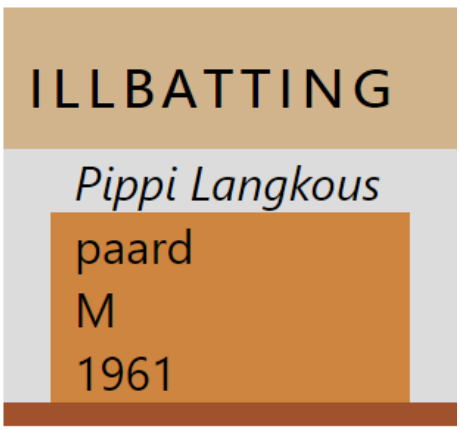
```
▼<praktijk>
  ▼<dier>
      <naam>Buck</naam>
      <klant>Al Bundy</klant>
      <soort>hond</soort>
      <geslacht>m</geslacht>
      <geboren>1983</geboren>
    </dier>
  ▼<dier>
      <naam>Illbatting</naam>
      <klant>Pippi</klant>
      <soort>paard</soort>
      <geslacht>m</geslacht>
      <geboren>1961</geboren>
    </dier>
</praktijk>
```

FIGUUR 1.8

Opdracht 6 XML III: opmaak

In XML wordt alleen data opgeslagen. Je kunt die data meer leesbaar maken door er vormgeving aan toe te voegen, zoals in figuur 1.9 voor één van de dieren uit de praktijk in figuur 1.7 is gedaan.

21. Open het XML-bestand *H1O06_dierenarts_1.1.xml* in je browser en bekijk het resultaat.
22. Open het bestand in een *editor*: op welke manier is er voor gezorgd dat de browser het bestand met opmaak laat zien?
23. Maak zelf een XML-bestand met de naam *H1O06_kleuren_1.xml* en verwerk hierin de data van figuur 1.4.
24. Voeg nu een bestand toe, zodanig dat jouw eigen xml-bestand met een nette vormgeving wordt weergegeven.



FIGUUR 1.9

Opdracht 7 data in een tabel

In de vorige opdrachten is een kleine dataset beschreven van dieren uit de praktijk van een dierenarts.

25. Maak een tabel van deze dataset, inclusief de gegeven metadata.
26. Wat is een record? Hoeveel records zitten er in deze dataset?
27. Waarom heet een relationele database *relationeel*? Wat is hier de betekenis van dat woord?
28. Leg uit waarom het beter is om het geboortejaar van een dier te registreren dan de leeftijd.
29. Je vindt de dieren in deze database terug op basis van hun naam. Leg uit waarom dit in een grote dierenpraktijk al gauw een probleem wordt.
30. Bedenk een oplossing voor het probleem uit de vorige vraag.

Opdracht 8 meerdere tabellen

In deze opgave bekijken we een uitgebreidere dataset van de administratie van een dierenarts in Excel. De dataset bestaat uit drie tabellen: *dieren*, *klanten* en *afspraken* (zie figuur 1.10).
Uit de vraagstelling van de vorige opdracht kun je concluderen dat er aan de dataset van figuur 1.7 nog wel zaken te verbeteren zijn.

datum	tijd	dier	notitie
3-01-22	10:00	2	eet slecht
3-01-22	11:00	9	
4-01-22	13:30	4	controle
4-01-22	10:30	6	castratie
4-01-22	13:30	1	sterilisatie
4-01-22	14:30	2	diabetes
7-01-22	10:00	4	

FIGUUR 1.10

31. Open het Excel-bestand *H1O08_dierenarts_2.0.xlsx*. Bekijk de inhoud van de drie tabbladen *dieren*, *klanten* en *afspraken*.
32. Hoe is het probleem waar vraag 29 naar verwijst hier opgelost?
33. In de tabel *afspraken* zie je in het eerste record in de kolom dier het nummer 2 staan. Hoe heet dat dier?
34. Wat is de woonplaats van dit dier?
35. In vraag 27 werd gevraagd waarom een relationele database zo heet. Leg met vraag 33 en 34 uit dat er nog een manier is waarop er een relatie in een relationele database kan zijn.

De manier van opslaan zorgt ervoor dat je vragen kunt beantwoorden op basis van data die in verschillende tabellen staan. Zo'n vraag is eigenlijk een zoekopdracht. Beantwoord de volgende vragen:

FIGUUR 1.11 ANGEL DE ENGELVIS

36. Welk dier is het jongst?
37. Welk levend dier is het oudst?
38. Hoeveel katten staan er in de database?
39. Hoeveel katten horen bij een klant uit Groningen?
40. Zijn er klanten die nog nooit een afspraak hebben gemaakt? Zo ja, wat is of zijn hun namen?

Opdracht 9 kattendata

De kunstenaar Owen Mundy heeft de website *I know where your cat lives* gemaakt, om te laten zien hoe eenvoudig het is om gegevens te verzamelen die mensen al dan niet bewust op internet achterlaten.

41. Klik op het icoontje hiernaast om naar de website te gaan.
42. Kun je een kat vinden in jouw buurt of van iemand die je kent?
43. Welke data is nodig om deze website te maken?

Mensen plaatsen vaak foto's van hun huisdieren op het internet met daarbij de naam van het dier.

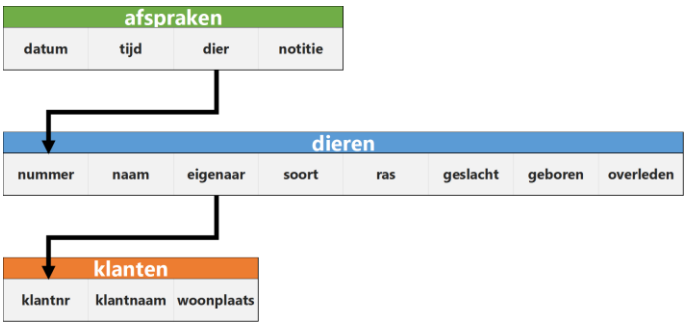
FIGUUR 1.12 BRIECK DE KAT

44. Leg uit hoe je met behulp van een foto de locatie (soms) kunt achterhalen.
45. Leg uit dat de gevonden locatie niet altijd klopt.

1.3 Databasemodellen

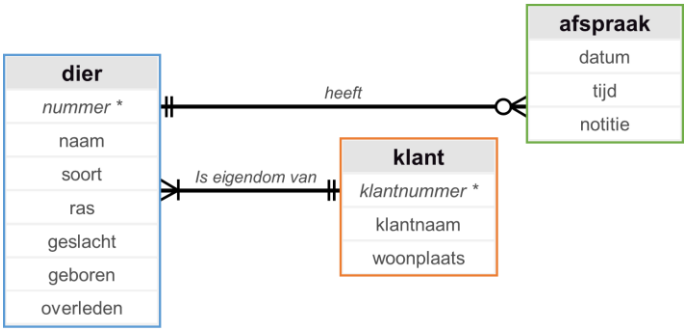
In de vorige paragraaf hebben we twee databasemodellen of -paradigma's gezien om data gestructureerd op te slaan. Als je een nieuwe database maakt, is de keuze voor een databasemodel zoals een **tabel** de eerste stap. Maar binnen dat databasemodel of -paradigma moet je nog steeds zelf structuur aanbrengen.

In figuur 1.13 is de database van de dierenartsenpraktijk uit opdracht 8 schematisch weergegeven in een **strokendiagram**. Merk op dat er geen data te zien is, maar alleen metadata. Het diagram toont de structuur waarbinnen de data wordt opgeslagen. Alle tabellen worden beschreven met daarbij de attributen van de datapunten van de records in die tabel. De zwarte pijlen in figuur 1.13 tonen de relaties tussen de tabellen van de database. Deze heten **referenties**.



FIGUUR 1.13

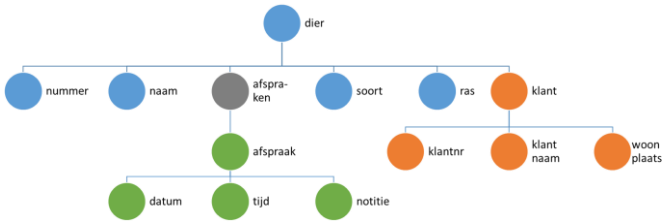
Als we dezelfde data van de dierenartsenpraktijk niet met tabellen maar met het **objectmodel** willen beschrijven, ontstaat een diagram zoals in figuur 1.14. De drie objecten worden beschreven op basis van een klasses met onderlinge relaties. Op de gebruikte symbolen bij die relaties komen we terug.



FIGUUR 1.14

De gegevens van een dier uit de objectdatabase vind je in figuur 1.14 terug door gebruik te maken van het attribuut *nummer*. Het *nummer* geeft je de toegang tot de bijbehorende data van het dier en wordt daarom de **primaire sleutel** genoemd. Een dier terugvinden lukt alleen als dit nummer **uniek** is. We zeggen dan dat voor het attribuut *nummer* **uniciteit** geldt.

In figuur 1.13 zien we een strokendiagram met drie tabellen. In het objectdiagram van figuur 1.14 zien we een diagram met drie objecten. Als we het dier centraal stellen, kunnen we zeggen dat een dier in de administratie van de dierenarts eigenschappen heeft, waaronder mogelijke afspraken en een eigenaar (klant). De klant heeft (net als b.v. een afspraak) zelf ook weer eigenschappen.



FIGUUR 1.15

In figuur 1.15 is dit getekend als een **boomstructuur** met meerdere vertakkingen die de hiërarchie tussen de elementen laat zien. Het beschrijven van data met een boomstructuur is een specifiek voorbeeld van het databasemodel **graaf** dat we in de opgaven verder zullen bekijken. In een boomstructuur of **boom** heten de elementen **datapunten** of **knopen**. Knopen zijn met elkaar verbonden door **takken**. Het bovenste datapunt heet de **wortel** van de boom: in figuur 1.15 is dat *dier*.

Bij het beschrijven van een graaf worden ook in Nederland vaak Engelse termen gebruikt. Een boom is een *tree* met *branches* (vertakkingen) die begint bij de *root*. Een element is een *node*. Een *node* die zich lager in de *tree* bevindt, heet een *child*: *klant* is een *child* van *dier*. Andersom is *dier* de *parent* van *klant*. Een link tussen twee *nodes* wordt behalve een *branch* ook een *edge* genoemd.

Opdracht 10 modellen voor een kleurendatabase

In deze opdracht kijken we naar een dataset met kleurennamen. Daarbij kijken we in eerste instantie naar een database op basis van een tabel.

46. Open het Excel-bestand *H1O10_kleuren_1.xlsx* en bekijk de data op het tabblad *kleuren*.
47. Zijn er attributen in de tabel waarvoor uniciteit geldt? Zo ja: welke?
48. Bekijk nu de data op het tabblad *kleuren_uitgebreid*. Zijn hier attributen met uniciteit?

Het attribuut *hex* op het tabblad *kleuren_uitgebreid* is berekend met de data in de kolommen B, C en D. De data in kolom E heet daarom **afgeleide data**. Om dat te benadrukken hebben we kolom E grijs gemaakt.

49. Noem een voordeel en een nadeel van het toevoegen van afgeleide data aan een database.

De data die je nu in het tabel-formaat ziet, kan ook met het objectmodel worden beschreven.

50. Wat is in die dataset dan de *primary key* of de primaire sleutel?
51. Beschrijf de data van *kleuren_uitgebreid* met het objectmodel, door een ontwerp voor een object te tekenen voor een kleur, vergelijkbaar met de aanpak in figuur 1.14.
52. Beschrijf de data ook met een graaf. Teken een boomstructuur (*tree*), op de manier van figuur 1.15.

Opdracht 11 formats en het ontbreken van data: null

Als je een database wilt maken, kies je eerst voor het databasemodel. Vervolgens beschrijf je de data en hun onderlinge relaties en denk je na over mogelijke beperkingen, zoals het format van de data:

FIGUUR 1.16

53. Leg uit dat data voor het attribuut *postcode* aan hele precieze eisen moet voldoen.
54. Noem nog drie voorbeelden van attributen waarvoor het verstandig is de inhoud qua format te beperken.
55. Leg uit dat een attribuut dat als primaire sleutel dient altijd een waarde moet hebben.

Het is voor een datapunt (*node*) of een attribuut van een *record* niet altijd nodig dat er een waarde is. Soms blijft een dataveld leeg.

56. Bekijk de video over het ontbreken van data.
57. Verklaar met behulp van de video wat *null island* is en waar op aarde het zich bevindt.
58. Leg in eigen woorden uit wat *null* betekent.

In de theorie bij deze paragraaf hebben we drie modellen gezien voor de database van een dierenarts.

59. Open (nogmaals!) het Excel-bestand *H1O08_dierenarts_2.0.xlsx* met de relationele database op basis van tabellen.
60. Zijn er velden met de waarde *null*? Hoe zie je dat?
61. Bekijk de tabel *afspraken*. Zijn er kolommen waarvan de bijbehorende waarde van het record *null* zou mogen blijven? Licht je antwoord toe.

In figuur 1.17 zie je een *screenshot* met twee objecten – de dieren Marsalis en Snowy – uit het administratieprogramma van de dierenarts op basis van het objectmodel.

FIGUUR 1.17

62. Vergelijk figuur 1.17 met figuur 1.14: welke data-attributen worden niet door het programma op het scherm getoond?
63. Bij Marsalis is er op twee manieren sprake van meer data dan bij Snowy. Welke twee manieren zijn dat?
64. Is voor beide van deze manieren sprake van een waarde *null*?

Opdracht 12 alternatieve boomstructuur

In de theorie bij deze paragraaf wordt een database voor de administratie van een dierenartsenpraktijk besproken. Stel dat jij zo'n database moet maken.

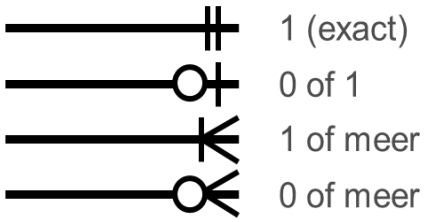
65. Wie stel je dan centraal? De *klant* of het *dier*? Geef minimaal één argument.

In figuur 1.15 is de database van de dierenarts als boom gegeven met *dier* als wortel (*root*) van de boom.

66. Teken een boom met *klant* als wortel.
67. Teken ook een boom met *afspraak* als wortel.
68. Is *afspraak* in jouw boom een *child* van *klant* of van *dier*?
69. Bekijk figuur 1.15 (en figuur 1.14): Is *afspraak* daarin een *child* van *klant* of van *dier*?
70. Zijn er elementen die in de ene boom een *parent* zijn van een ander *element* terwijl ze in de andere boom juist een *child* van dat element zijn?
71. Wat is volgens jou de beste keuze als wortel van de boom? Is dat *dier*, *klant* of *afspraak*? Waarom?

Opdracht 13 kardinaliteit I: soorten relaties

In de theorie van deze paragraaf worden relaties beschreven tussen tabellen, objecten en datapunten. Het aantal relaties kun je beschrijven op basis van de **kardinaliteit** van de relatie. Dit kan worden gedaan met de *kraaienpootnotatie* in figuur 1.18.



FIGUUR 1.18

Eerst een voorbeeld:
Een dier hoort (in de administratie) altijd bij één klant, maar een klant kan meerdere dieren hebben. In figuur 1.14 is dit aangegeven met symbolen, waarvan je in figuur 1.18 de bijbehorende betekenis ziet.

Kardinaliteit geeft aan hoeveel objecten van de ene soort gekoppeld kunnen zijn aan hoeveel objecten van de andere soort. Of voor tabellen: hoeveel records in de ene tabel een relatie kunnen hebben met hoeveel records uit een andere tabel.

Beantwoord de volgende vragen met behulp van figuur 1.14 en figuur 1.18:

72. Hoe zie je dat een dier niet gekoppeld kan worden aan meerdere klanten?
73. Is het mogelijk om een straatkat (zonder eigenaar) in de administratie op te nemen? Hoe zie je dat?
74. Hierboven (bij het voorbeeld) staat de relatie tussen *dier* en *klant* in een Nederlandse zin uitgelegd. Leg de relatie tussen *dier* en *afspraak* uit met een vergelijkbare zin.
75. Verklaar dat het bovenste symbool in figuur 1.18 alleen kan worden toegepast op objecten die een attribuut hebben met uniciteit.

☆ Opdracht 14 XML IV: XML versus HTML

Als je zelf al eens een website hebt gemaakt, is het je vast opgevallen dat XML-documenten erg lijken op HTML-documenten? Data wordt in XML beschreven met **elementen** zoals `<naam>Angel</naam>` die zijn gemaakt met een *start-tag* (`<naam>`), een *eind-tag* (`</naam>`) en de data (**Angel**). De elementen zijn **genest**: elementen bevinden zich binnen andere elementen (die soms zelf weer elementen bevatten).

HTML bevat ook elementen, maar deze zeggen niet perse iets over de data, maar iets over de rol in het document. In HTML is `<h1>Angel</h1>` een kop (van niveau 1; Engels: *header*). Dit vertelt niet dat het een naam is.

76. Open het XML-bestand *H1O14_klas_1.0.xml* in je browser en bekijk de geneste elementen.
77. Maak een boomstructuur bij de data in dit XML-document.
78. Open het HTML-bestand *H1O14_klas_1.1.html* in je browser en bekijk het resultaat.
79. Open nu het HTML-bestand *H1O14_klas_1.1.html* in je editor en bestudeer de HTML-code met haar geneste elementen.
80. Maak een boomstructuur bij de data in dit HTML-document.