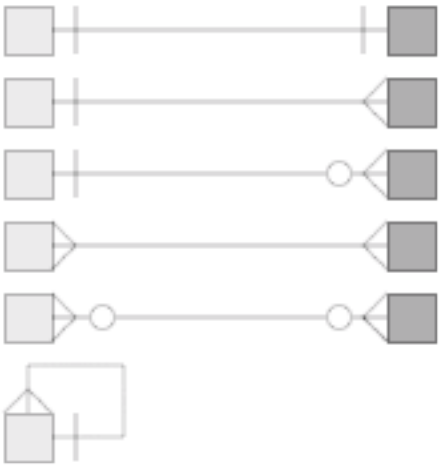
INHOUDSOPGAVE

1 Entiteit-Relatie Diagram

DataModellering heeft tot doel het foutpercentage tijdens software-ontwikkeling terug te brengen,de programmeertijd te bekorten en het te automatiseren onderwerp zo optimaal mogelijk weer te geven.

De bij het modelleren te hanteren schematechniek zijn de Entiteit/Relatie diagrammen, daar deze techniek bij uitstek geschikt is om helder en duidelijk de relaties tussen de verschillende entiteiten weer te geven. Hieronder volgen enige voorbeelden van mogelijke relaties en de weergave daarvan middels deze techniek:



Een 1:1 relatie

Een 1:n relatie

Een 1:n of 1:0 relatie

Een m:n relatie

Een m:n en m:0 en 0:n relatie

Een 1:n relatie van dezelfde entiteit. ([mens] ouders-> kinderen)

2 Normaalvormen

Om tot een goed gegevensmodel te komen zijn er regels en technieken opgesteld die overal, waar ontworpen wordt, worden toegepast.

Elk stuk relevante informatie binnen de organisatie, of naar de organisatie toe, moet worden geregistreerd. Elk stuk informatie mag slechts één keer voorkomen binnen de geregistreerde informatie. Om dit te bewerkstelligen maakt men gebruik van een wereldwijd aanvaarde techniek, normalisatie. Deze techniek kent drie fasen, te weten:

2.1 De 1e normaalvorm

- Herhalende groepen uit huidige entiteiten zoveel mogelijk naar nieuwe entiteiten.
- De afhankelijkheid van attributen is in deze fase nog niet interessant.
- Als resultaat een concept gegevensmodel met alle entiteiten en bijbehorende attributen, weergegeven als tabellen, waarvan één attribuut is gekozen waarmee de informatie binnen de tabel uniek identificeerbaar is.

2.2 De 2e normaalvorm

- Voldoet aan de eerste normaalvorm.
- Alle attributen zijn volkomen afhankelijk van de unieke sleutel.
- Als resultaat nieuwe tabellen, gevormd uit reduntante gegevens, met weer één attribuut waarmee de informatie binnen de nieuwe tabel uniek identificeerbaar is.

2.3 De 3e normaalvorm

- Voldoet aan de tweede normaalvorm.
- Alle attributen mogen niet transitief afhankelijk zijn van de unieke sleutel. Dat betekent dat attributen binnen één entiteit niet afhankelijk van elkaar kunnen en mogen zijn.

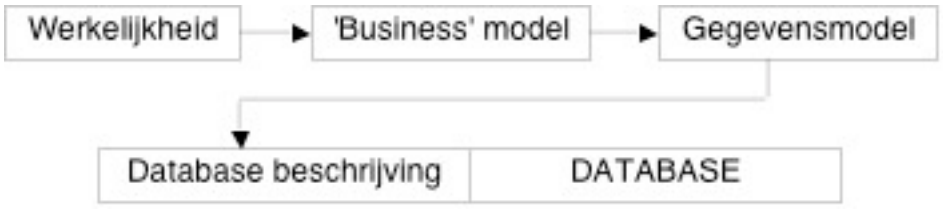
Er mogen geen onderling onverenigbare representaties van informatie voorkomen, noch conflicterende regels met betrekking tot wat met de informatie gedaan kan worden.

Er moet een duidelijk onderscheid bestaan tussen de informatie die wordt gepresenteerd en de eigenschappen van de te presenteren informatie.

- Een exacte specificatie van de eigenschappen van de te presenteren informatie die gebruikt zullen worden om de informatie uniek identificaarbaar te maken.
- Een explicite definitie van de relaties tussen de te presenteren stukken informatie.

Naast het gegevensmodel dienen er ook manieren bepaald te worden om verklaringen weer te geven:

- Een wijze waarop niet-procedurele regels om de integriteit van het model te waarborgen kunnen worden gedefineerd.
- Een wijze waarop eenvoudige procedurele regels, die deel uit maken van het model, kunnen worden gedefinieerd.
- Een wijze waarmee geregistreerd kan worden waarom de interpretatie van de werkelijkheid heeft geleid tot het model en waarom juist déze interpretatie nodig is:



2.4 Werkelijkheden

We kennen drie werkelijkheden. Een stuk informatie valt altijd in één van de drie, afhankelijk van het type informatie. Als we het over datamodellen hebben, hebben we het impliciet over een interpretatie van een deel van de werkelijkheid zoals wij die kennen. Het is dan dus duidelijk dat een model afhankelijk is van de belevingswereld van de ontwerper van het model.

- Al het tastbare.
  (producten, hardware, etc.)

- Al het niet-tastbare.
  (gedachten van werknemers, klanten, etc.)

- Al het niet-tastbare dat omgezet kan worden naar het tastbare.
  (werkwijzen van werknemers, werkwijzen binnen de organisate, etc.)

  Hier onder volgen enige voorbeelden:

  - Een bedrijf of organisatie valt in {3}, de objectieve werkelijkheid.
  - Een gegevensmodel valt in {3}.
  - Implementatie van een model wordt een {2}-proces op {1}-hardware.
  - Een pakbon valt in {1} en een order in {3}.
  - Een prijslijst valt in {1} en een prijs in {3}.

Elk model zal beperkingen en vereenvoudigingen inhouden ten opzichte van de werkelijkheid.

- Het proces waarmee de organisatie en het model worden bekeken is subjectief. Er zijn namelijk veel meer interpretaties mogelijk, doch het éne model benaderd de perceptie van de ontwerper, met betrekking tot de organisatie, meer dan het andere model.

- Het kan mogelijk zijn, soms te wensen, dat de organisatie zich aan dient te passen om vereenvoudiging van de werkelijkheid te bewerkstelligen ten behoeve van het te creëren gegevensmodel.

- Probeer nooit alles te automatiseren. Leg duidelijke grenzen tussen de te automatiseren processen en processen die beter, of eenvoudiger, binnen de organisatie opgenomen, of opgelost, kunnen worden.

- Bepaal welke processen met reeds bestaande programmatuur of met standaard pakketten opgelost kunnen worden. Denk aan een voudig lijstwerk middels een kantoorautomatiseringsconcept (SQL -> Tekstverwerker).

Een entiteit is iets waar we informatie van wensen te registreren en onderhouden en is identificeerbaar via één of meerdere attributen. Het is een categorisatie van dingen die onder één noemer vallen. Bijvoorbeeld 'VOLVO', 'CITROËN' en 'MITSUBISHI' kunnen alle onder de noemer 'AUTO' vallen.

Het kiezen van entiteiten is een belangrijke stap na de normalisatie. Je zou deze fase zelfs de 4e normaalvorm kunnen geven. Het gaat hier om het samenvoegen of juist splitsen van entiteiten omdat ze óf in dezelfde categorie vallen (samenvoe gen) óf dat er juist verschillende klassen binnen één categorie voorkomen (splitsen).

- Als blijkt dat twee entiteiten bekeken kunnen worden als zijnde van dezelfde categorie (auto/vrachtauto) dan zouden ze samengevoegd kúnnen worden tot één entiteit.

- Als blijkt dat er binnen één entiteit twee verschillende zaken worden geregistreerd, kán besloten worden de entiteit te splitsen (uitgevers/schrijvers in 'makers van boeken').

3 Modelleren

Creer een lijst van alle relevante entiteiten.

- De entititeit ORDER, bijvoorbeeld, is relevant, de entiteit ORDERREGEL nog niet. De orderregels vormen een repeterende groep binnen ORDER en deze wordt bij fase 4 uitgenormaliseerd. In deze fase neem je alle attributen die je kan verzinnen en betrekking hebben op een order op bij ORDER.

Creer een lijst van attributen die als kandidaat-sleutel kunnen fungeren om de entiteiten uniek identificeerbaar te maken. Leg nog géén relaties tussen entiteiten. Dus niet bij ORDER het ORDERNR en KLANTCODE als sleutel want klantcode heeft een relatie met KLANT. Het attribuut KLANTCODE ontstaat vanzelf verderop in proces.

- Let op de grootte van een sleutel. Men mag, als er geen goede sleutel voorhanden is nieuwe sleutels introduceren, bijvoorbeeld KLANTCODE in plaats van KLANTNAAM.

- Het kan zijn dat bepaalde entiteiten niet uniek ideentificeerbaar zijn. Probeer ze dan géén sleutel toe te kennen. Deze kan bij fase 5 naar voren komen.

Creer per entiteit een lijst met in de entiteit op te nemen relevante gegevens, objectieve èn subjectieve.

Pas de normalisatie techniek toe.

- Let tijdens het normaliseren op grote hoeveelheden informatie die niet altijd aanwezig hoeft te zijn. Daarmee doelen we op informatie die er gewoon niet altijd is en op informatie binnen een entiteit die, bekeken vanuit de relaties van die entiteit, slechts bij erg weinig relaties van belang is. In deze gevallen kan het verstandig zijn (ruimte besparing en snelheid) om voor díe gegevens die nauwelijks aanwezig zijn óf nauwelijks met andere entiteiten worden gedeeld een nieuwe entiteit te creeren met dezelfde sleutel als de entiteit waar ze oorspronkelijk waren opgenomen. Een

financieel model, bijvoorbeeld, is misschien geïnteresseerd in een paar attributen van entiteit KLANT, maar de orderverwerking juist weer in bijna alle.

Leg de relaties tussen de diverse entiteiten. In deze fase zullen er attributen aan de verschillende entiteiten worden toegevoegd om de relatie te bewerkstelligen. Dit mogen attributen zijn die samen met de sleutel van het attribuut een vervangende unieke sleutel gaan vormen of gewone attributen waarmee enkel de relatie wordt vastgelegd.

- Indien de relatie een 1:n relatie weergeeft op dezelfde entiteit (Ouders/Kinderen in MENS), verklaar deze relatie met maximaal twee regels tekst.

- Als mocht blijken dat gegevens via-via nodig zijn en men, gevoelsmatig al, problemen heeft met het opnemen van een relatie in de betreffende entiteit (bijvoorbeeld voor slechts één andere entiteit of het kán gewoon niet) geef dan door middel van een procedurele regel weer hoe aan de benodigde informatie te komen is. Men heeft binnen een order bijvoorbeeld de laatst geldende prijs van een artikel nodig. Dan haalt je via de entiteit ARTIKEL de prijs op met als extra parameter de datum om aan de laatst geldende prijs te komen:

  - Let wel, in 3GL is alles op te lossen.
  - Niet alle databasetools kunnen deze oplossing in hun datadictionairy kwijt. Het vereist dan dus extra programmeerwerk!

Bepaal voor alle attributen het type, dus (alfa)numeriek, drijvende komma (met het aantal decimalen), packed, etc. en de lengte. Met deze gegevens kan men de grootte van één record berekenen en aan de hand van de kwantiteiten die uit het vooronderzoek naar voren zijn gekomen, kan men de te verwachten bestandsgrootte uitrekenen zodat men rekening kan houden met de benodigde schijfcapaciteit.

Creer een lijst met voor elk attribuut de grenswaarden, indien van toepassing, met de bijbehorende foutmeldingen en uitgangen indien niet aan de voorwaarde(n) kan worden voldaan, bijvoorbeeld:

- (Alfa)numerieke grenswaarden. (0 - 999 of J - N)
- Conditionele grenswaarden. (Als postkode is gevulddan móet huisnr > 0 zijn)
- ...

Bepaal, als laatste, de alternatieve sleutels, bijvoorbeeld de postkode in entiteit KLANT. Bepaal of de index fysiek of dynamisch moet zijn. Fysieke opslag betekent méér schijfruimte, méér database I/O tijdens onderhoud en daarmee vertraging voor alle toepassingen die de betreffende entiteit kunnen en mogen onderhouden. Dynamische creatie betekent tijdelijke opslag en daarmee vertraging voor díe functies die via deze index de tabel benaderen.

- Bij sommige 4GL-tools wordt de index met de unieke sleutel dynamisch opgebouwd indien men met de entiteit werkt (de opslag is geregeld middels een zogenaamde 'hash-table') en de alternatieve sleutels worden in fysieke indexen opgenomen. Bij sommige 4GL-tools is het net andersom.

4 Voorbeeld

Op de volgende pagina's zullen we een voorbeeld uitwerken. Het voorbeeld betreft een paardenrace administratie en is deels ontleend aan de Synon/2 handboeken.

We onderkennen de volgende entiteiten:

- Paard
- Bereider
- Wedstrijd

Vervolgens stellen we per entiteit de gewenste attributen vast:

Paard
- Naam
- Geslacht
- Waarde
- Geboortedatum
- Eindpositie
- Entree-nummer
- Handicap
- Wedstrijd-status
- Aantal gewonnen wedstrijden
- Tot nu toe gewonnen prijzengeld

Bereider
- Naam
- Geslacht

Wedstrijd
- Naam
- Aanvangsdatum
- Starttijd
- Condities
- Afstand
- Prijzengeld
- Welke baan

De volgende stap is het normalisatieproces. Als eerste gaan we herhalende groepen afzonderen en overbrengen naar nieuwe entiteiten. Bekijk de entiteit PAARD eens goed en zie dan dat er gegevens zijn die we eigenlijk maar één keer nodig hebben, zoals bijvoorbeeld de naam van het paard, en gegevens die vaker voor zullen komen. Díe gegevens die vaker voor zullen komen noemen we een repeterende groep, of 'multi-occurrence'. Voor deze gegevens definiëren we een nieuwe entiteit, PAARDHIS(torie). Dit houden we over:

Paard
- Naam
- Geslacht
- Waarde
- Geboortedatum
- Tot nu toe gewonnen prijzengeld

Paardhis
- Naam
- Eindpositie
- Entree-nummer
- Handicap
- Wedstrijd-status

Om de nieuwe entiteit aan één PAARD te koppelen is de naam van het paard meegnomen. Hieruit volgde dat we het 'Aantal gewonnen wedstrijden' kunnen tellen uit PAARDHIS door de eindpositie te testen van alle recordtypen waarvan de naam hetzelfde is als de naam uit PAARD. Dit totaalveld is dus uit PAARD verdwenen. We hebben automatisch, omdat er geen andere attributen voor inaanmerking kwamen, de naam kandidaat sleutel gemaakt. Dan bedenken we ons dat een naam als sleutel in de toekomst veel beslag gaat leggen op het systeem (schijfruimte, méér I/O) en ook niet uniek is, dus introduceren we een sleutel die we uniek kunnen maken. We geven elk

paard een, binen deze entiteit, uniek nummer en via dit nummer zullen de andere tabellen aan PAARD kunnen refereren:

Paard
- **Code**
- Naam
- Geslacht
- Waarde
- Geboortedatum
- Tot nu toe gwonnen prijzengeld (totaal)

Paardhis
- **Code**
- Eindpositie
- Entree-nummer
- Handicap
- Wedstrijd-status

Hetzelfde doen we bij BEREIDER. Een naam als sleutel neemt veel te veel ruimte in:

Bereider
- **Code**
- Naam
- Geslacht

Vervolgens nemen we de volgende entiteit onder handen, WEDSTRIJD. Na normalisatie houden we het volgende over:

Wedstrijd
- **Code**
- Naam

Wedstrijdplaats
- **Code**
- Aanvangsdatum
- Starttijd
- Condities
- Afstand
- Prijzengeld
- Welke baan

Er bestond een repeterdende groep binnen WEDSTRIJD en voor die repeterende groep is een nieuwe entiteit gecreerd, WEDSTRIJDPLAATS. Tevens is de wedstrijd uniek identificeerbaar gemaakt door een nieuwe attribuut toe te voegen en dat tegelijkertijd als sleutel te definiëren.

De volgende stap in het proces is het leggen van de relaties tussen de entiteiten. Dat betekent dat men na gaat denken over de informatie die men niet binnen één entiteit tot beshikking heeft doch wel als één geheel wenst te presenteren, cq. te onderhouden. We hebben nu de volgende entiteiten:

- Paard
- Paardhis
- Wedstrijd
- Wedstrijdplaats
- Bereider

Wat willen we nu zien:

- Wie zijn de ouders van een paard?
- Welk paard heeft welke wedstrijden gereden?
- Bij welke wedstrijd hebben welke paarden gelopen?
- Welke bereider heeft welke paarden bereden?
- Welke bereider heeft welke wedstrijden gereden?

- Hoeveel prijzengeld heeft een paard binnengehaald?
- Hoevaak heeft een paard een wedstrijd gewonnen?
- etc...

Om de eerste vier vragen te kunnen beantwoorden dienen er relaties tussen de verschillende entiteiten te liggen:

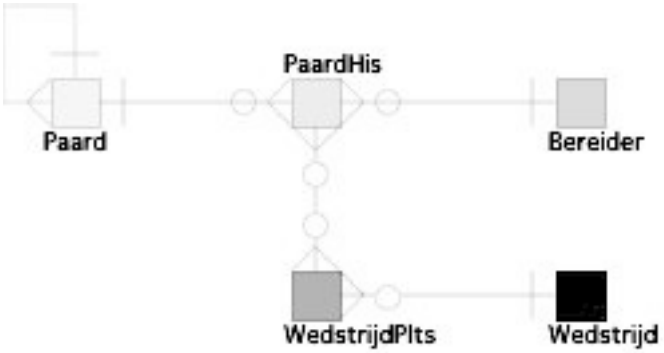
- Binnen PAARD zelf bestaat een 1:n relatie. Eén paard heeft een relatie met meerdere paarden (ouders).

- Eén PAARD heeft meerdere relaties met zijn historie. Het heeft geen, één of meerdere wedstrijden gereden.

- Eén PAARD heeft een relatie met één of meerdere bereiders. Indien er geen wedstrijden gereden zijn, zijn er ook geen bereiders. Dat betekent dat we de relatie moeten leggen via de historie van een paard. Daardoor veranderd de relatie: één historierecord heeft een relatie met één bereider, doch een bereider heeft relaties met nul, één of meerdere historierecords. Tevens kunnen we de relatie WEDSTRIJD <-> PAARD via deze tabel realiseren, want zodra er een historierecord bestaat is er een wedstrijd gereden.

- Eén WEDSTRIJD kan op meerdere plaatsen worden gereden en op één plaats kunnen meerdere wedstrijden worden gehouden.

- Het totale prijzengeld kan geteld worden via PAARDHIS, en dit kan dus uit PAARD verdwijnen.
-



Voor de entiteiten houdt dat het volgende in:

- Bij PAARD moeten twee attributen opgenomen worden waarvan één refereert aan de moeder van het paard en de andere aan de vader refereert. De entiteit PAARD komt er, definitief, als volgt uit te zien:

  Paard
  - **Code**
  - ...
  - *Code Moeder*
  - *Code Vader*

- De volgende is WEDSTRIJDPLAATS. We hebben geconstateerd dat een wedstrijd op meerdere plaatsen gereden kan zijn, doch ze zal nooit op meer dan één plaats op één

dag gehouden worden. Om deze 1:n relatie met wedstrijd te bewerkstelligen, moeten we de aanvangsdatum opnemen in de sleutel:

Wedstrijdplaats	• **Code** • ***Aanvangsdatum*** • ...

- Nu is PAARDHIS aan de beurt. De relatie met PAARD is nog 1:1, doch dat gaan we nu veranderen. Om te kunnen achterhalen welke wedstrijden het paard ge reden heeft, moet er een 1:n relatie liggen met WEDSTRIJDPLAATS. Deze wordt gelegd via PAARDHIS. Daartoe nemen we de sleutel van WEDSTRIJDPLAATS op als referentie naar WEDSTRIJDPLAATS. Daarmee creeren we een nieuwe, maar unieke, sleutel. Maar we zijn nog niet klaar. Hoe kom je te weten welke jockey welk paard heeft gereden en bij welke wedstrijd? Door de sleutel van BEREIDER op te nemen als referentie. PAARDHIS komt er uiteindelijk als volgt uit te zien:

Paardhis	• **Code** • ***Code wedstrijd*** • ***Aanvangsdatum*** • ... • Bereider

Het model ligt nu vast. Je kan uiteraard nog veel verder gaan, doch dit is slechts een voorbeeld. De volgende stap is het bepalen van de attribuuttypen. Daar bedoelen we mee dat we bepalen of een attribuut (alfa)numeriek, etc. is en de lengte van het attribuut. Hieronder volgt een mogelijke oplossing:

Paard	• **Code**	Alfanumeriek	4 pos.
	• Naam	Alfanumeriek	4 pos.
	• Geslacht	Boolean	
	• Waarde	Drijvende komma	2 decimalen
	• Geboortedatum	Datum	
	• Code Moeder		
	• Code Vader		
Bereider	• **Code**	Alfanumeriek	2 pos.
	• Naam	Alfanumeriek	30 pos.
	• Geslacht	Boolean	
Wedstrijd	• **Code**	Alfanumeriek	4 pos.
	• Naam	Alfanumeriek	30 pos.
Wedstrijdplaats	• **Code**	Alfanumeriek	4 pos.
	• **Aanvangsdatum**	Datum	
	• Starttijd	Tijd	
	• Condities	Alfanumeriek	1 pos.
	• Afstand	Numeriek	5 pos.
	• Prijzengeld	Drijvende komma	2 decimalen
	• Welke baan	Alfanumeriek	1 pos.

Paardhis	• **Code**	Alfanumeriek	4 pos.
	• **Code wedstrijd**		
	• **Aanvangsdatum**		
	• Eindpositie	Numeriek	2 pos.
	• Entree-nummer	Numeriek	2 pos.
	• Handicap	Numeriek	2 pos.
	• Wedstrijd-status	Alfanumeriek	1 pos.
	• Bereider		

De cursief gedrukte velden refereren aan attributen in een andere entiteit en krijgen dus automatisch het type en de lengte van het attribuut waarnaar ze wijzen.

We gaan nu de laatste fase in, het weergeven van de condities en grenswaarden van de genoemde attributen:

Paard	• Code	1111 t/m 9999
	• Naam	minimaal 5 tekens
	• Geslacht	M of V
	• Waarde	Groter of gelijk aan 0,00
	• Geboortedatum	Nul (onbekend) of een geldige datum. Als de ouders bekend zijn moet de datum kleiner zijn dan de geboortedatum van de ouders.
	• Code Moeder	Moet een *paard* zijn met geslacht 'V' en een geboorte-datum groter dan die van het paard.
	• Code Vader	Moet een *paard* zijn met geslacht 'M' en een geboorte-datum groter dan die van het paard.
Bereider	• Code	11 t/m 99
	• Naam	minimaal 5 tekens
	• Geslacht	M of V
Wedstrijd	• Code	1111 t/m 9999
	• Naam	minimaal 5 tekens
Wedstrijdplaats	• Code	Een bestaande *wedstrijd*
	• Aanvangsdatum	Een geldige datum
	• Starttijd	Een geldige tijd
	• Condities	1 t/m 9, A t/m Z
	• Afstand	Groter dan nul
	• Prijzengeld	Groter of gelijk aan 0,00
	• Welke baan	1 t/m 9, A t/m Z
Paardhis	• Code	Bestaand *paard*.
	• Code wedstrijd	Bestaande *wedstrijdplaats*.

• Aanvangsdatum	Bestaande *wedstrijdplaats*.
• Eindpositie	00 t/m 99
• Entree-nummer	00 t/m 99
• Handicap	00 t/m 99
• Wedstrijd-status	A t/m Z D : Gediskwalificeerd E : Geïndigd N : Nog niet gelopen W : Gewond ... : ...
• Bereider	Een bestaande *bereider*

Dit zijn slechts voorbeelden van mogelijke grenswaarden.

Om het gegevensmodel op meer ingangen te benaderen dan via de unieke sleutels, kunnen we alternatieve sleutels definiëren. Een alternatieve sleutel hoeft niet uniek te zijn en wordt gekozen uit de overige attributen van de entiteit. Stel dat we PAARDHIS ook willen benaderen via de wedstrijdcode. Zoals de entiteit nu is opgebouwd, betekent dat de software sequentieel door het bestand heen gaat, omdat het betreffende veld zelf géén sleutel is en dus niet in een bepaalde volgorde ligt:

Via de hoofdsleutel

Paardcode	Wedstrijdcode	Aanvangsdatum
0001	9001	15-01-1990
0002	8903	10-03-1989
0003	9006	05-06-1990
0004	9001	15-01-1990
...	...	...

Als we nu in PAARDHIS alle records met wedstrijdcode 9001 willen hebben, moeten we dus vier records lezen. Dat kost tijd als het bestand groot is, wat met historische bestanden altijd het geval is. We creëren dus een tweede ingang op wedstrijdcode. We hoeven dan slechts twee records te lezen. Een winst van 50% tegen iets meer tijd tijdens het onderhouden van het bestand, want er moet een index extra bijgewerkt worden:

Via een alternatieve sleutel

Wedstrijdcode
8903
9001
9001
9006
...

Door middel van indexering wordt met lezen gestart op record 2 en vinden we gelijk een record met wedstrijdcode 9001. Vanaf dit punt wordt sequentieel verder gelezen totdat de gevonden sleutel ongelijk is aan de gevraagde sleutel.

Een andere, voor de hand liggende, alternatieve sleutel is *bereider* in PAARDHIS. Hier gelden precies dezelfde regels als bij de hierboven beschreven wedstrijdcode.

We hebben bij de index op wedstrijdcode aangenomen dat deze fysiek aanwezig is. De ontwerper had deze sleutel ook als dynamisch kunnen classificeren. Aan een dynamische index hangt meestal een conditie. Er wordt een subset gemaakt, door een DBMS of door een door de 4GL tool opgebouwde query, met alleen records die aan de conditie voldoen en die subset wordt aan de software aangeboden. De keuze tussen een fysieke en een dynamische index wordt vooral bepaald door het verwachte intensieve gebruik van de index en door het aantal fysieke indexen die reeds rond de entiteit zijn gedefinieerd.

In ons model geven we aan dat er twee dynamische indexen zijn, te weten een index op het geslacht van het PAARD voor de code waarin de vader wordt opgenomen met als conditie dat het geslacht een M moet zijn en een index voor de code waarin de moeder wordt opgenomen met als conditie dat het geslacht een V moet zijn.

Zoals je ziet komt er best nog wel wat denkwerk om de hoek kijken om een model in één keer zo goed mogelijk op te zetten. Als je denkt klaar te zijn, loop het model dan nog eens van begin af aan door, stap voor stap, en vraag je, bij elk idee dat je krijgt, af of het zinvol is. Ga er nooit vanuit dat je niets bent vergeten.

Denk aan de wetten van Murphy !