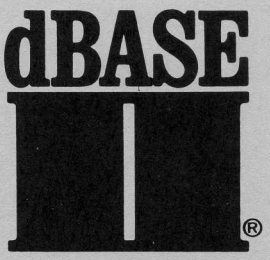


Everyman's Database Primer

Door
Robert A. Byers

ASHTON·TATE™
Postbus 71876, 1008 EB Amsterdam
Telefoon 020-462515

dBASE II®
HANDLEIDING VOOR DE GEBRUIKER

BELANGRIJKE INFORMATIE BETREFFENDE HET GEBRUIK VAN dBASE II EN VAN DEZE HANDLEIDING

Het is niet toegestaan om het computerprogramma dBASE II en/of de daarbij behorende handleiding op enige wijze te vermenigvuldigen, openbaar te maken, over te brengen op een elektronische gegevensdrager of op andere wijze in een op machinaal leesbare vorm om te zetten zonder uitdrukkelijke en schriftelijke toestemming van Ashton-Tate B.V.
Het zonder toestemming van Ashton-Tate B.V. vermenigvuldigen van het computerprogramma en/of de daarbij behorende handleiding is op grond van de auteurswet (waaronder begrepen eventuele persoonlijkheidsrechten) en overige wetgeving met betrekking tot bescherming van industriële eigendom, verboden. Bij overtreding van betreffende bepalingen van de wet zal door Ashton-Tate B.V. steeds in een civiele procedure schadevergoeding worden gevorderd en zal tevens aangifte worden gedaan bij de Officier van Justitie waarop een strafvervolging van overtreders kan plaatsvinden.
In een door de gebruiker en namens Ashton-Tate B.V. ondertekende software-licentie-overeenkomst zijn rechten en plichten van partijen vastgelegd.
Deze handleiding is bijgewerkt en aangevuld tot de dag van verzending en wordt in die vorm aan de gebruiker verstrekt.
Behalve de garantie(s) die in de software-licentie-overeenkomst zijn opgenomen wordt door Ashton-Tate B.V. geen enkele andere garantie gegeven.
Ashton-Tate B.V. verleent uitdrukkelijk geen garantie voor de verkoopbaarheid van het produkt of voor de geschiktheid voor een bepaald doel.

Ashton-Tate B.V. zal in geen geval aansprakelijk zijn voor enige directe of indirecte vorm van schade of verlies door een afnemer geleden, zelfs niet wanneer de afnemer Ashton-Tate B.V. van te voren op de hoogte heeft gesteld dat schade of verlies zal gaan ontstaan. (Voorzover beperking of uitsluiting van schade door de wet is beperkt of uitgesloten, is deze bepaling niet van toepassing.)

Printed in The Netherlands

Everyman's Database Primer
Een inleiding tot database management technieken met speciale aandacht voor dBASE II

Door
Robert A. Byers

Redactie
Virginia Bare

Uitgave
Ashton-Tate
Postbus 71876
1008 EB AMSTERDAM
020-462515

Postbus 71876
1008 EB AMSTERDAM
020-462515

ISBN 0-912677-00-7

Oorspronkelijke titel: Everyman's Database Primer featuring dBASE II.

Eerste druk mei 1985

INHOUD

Een handboek voor de "niet-computerdeskundige": van aanzetten van de computer, tot planning, opbouw, bijwerken en gebruiken van een relationeel database management systeem op een microcomputer.

Deze inleiding tot databases op microcomputers legt het gebruik van geavanceerd ondersteuningsgereedschap voor de microcomputer in handen van bijna elke gebruiker die er belangstelling voor heeft.

Geschreven door Robert A. Byers, Manager Mission Support Systems Implementation, Jet Propulsion Laboratory, Pasadena, Californië.

Een uitgave van Ashton-Tate

Ashton-Tate
Postbus 71876, 1008 EB AMSTERDAM, telefoon 020-462515

ISBN 0-912677-00-7

DEEL EEN

Deel Een stelt 'databases' aan u voor en laat u kennismaken met de werking ervan. Databases zijn heel gewoon in onzė dagelijkse omgeving en we denken er nauwelijks over na bij wat we zoal doen. Een 'database management systeem' is gewoon de manier waarop deze vertrouwde databases in een computer zijn opgeslagen en waarop ze door een computer bewerkt worden.

We zullen een paar voorbeelden van databases eens nauwkeurig bekijken. Zo kunnen we een database in zijn onderdelen ontleden en begrijpen, hoe een database systeem te werk gaat, als het ons geordende informatie verschaft. Dit gaat zo eenvoudig: we beginnen meteen maar, we zetten onze computer aan en maken onze eerste computerdatabases.

1. DATABASES

'Database' is een uitdrukking uit het computerjargon voor iets dat ons in het dagelijks leven heel vertrouwd is, en dat eigenlijk ook onmisbaar is. Een database is een hoeveelheid informatie die geordend is en die wordt weergegeven in een vorm die een bepaald doel dient.

Een heel vertrouwd voorbeeld van een database is het telefoonboek. Deze alledaagse database in gedrukte vorm vermeldt de namen, adressen en telefoonnummers van privé-personen, bedrijven en overheidsinstellingen. Op zichzelf hebben de adressen en telefoonnummers weinig waarde. Ze zijn alleen nuttig, wanneer ze worden vermeld in VERBAND met een bepaalde naam.

Goed beschouwd is het aantal databases waarmee we vertrouwd zijn, werkelijk verbazingwekkend. Een paar gewone databases zijn: het woordenboek, het kookboek, de Wehkamp catalogus, de encyclopedie, de catalogus-kaartjes in de bibliotheek, uw giromap, enzovoorts. Afhankelijk van uw achtergrond zijn voor u de volgende databases meer of minder vertrouwd: de beurskoersen in de krant, het crediteurenboek, de personeelslijst enzovoorts.

Waarom noemen we nu deze voorbeelden databases? Waarom wordt de krant of een roman niet als database beschouwd? Daar is een heel bepaalde reden voor. In alle bovenstaande voorbeelden wordt de informatie aangeboden op een manier die het voor de gebruiker gemakkelijk maakt een bepaald stuk informatie op te zoeken, dat van belang is. In het telefoonboek worden de telefoonnummers en de adressen bijvoorbeeld in verband gebracht met de naam. De namen worden in alfabetische volgorde aangeboden zodat u ze gemakkelijk kunt vinden. Zoek de naam op en u hebt het telefoonnummer gevonden. De naam is het TREFWOORD bij het gebruik van het telefoonboek. Bij het voorbeeld van het woordenboek is het net zo. Daar hebt u een woord en een omschrijving. De woorden staan alfabetisch zodat u ze gemakkelijk kunt vinden. De definitie staat in verband met het woord. Het TREFWOORD bij het gebruik van het woordenboek is het woord zelf.

Het gemeenschappelijke in alle voorbeelden is *georganiseerde informatie weergegeven op een manier waarop die informatie gemakkelijk terug te vinden is* – door gebruik te maken van een of ander TREFWOORD. Met andere woorden, informatie die kan worden weergegeven in de vorm van een tabel (rijen en kolommen) kan een database zijn. In Figuur 1-1 ziet u een paar voorbeelden van kolomopschriften in tabellen die als databases beschouwd zouden kunnen worden.

Voorbeelden:	***kolomopschriften***				
TELEFOONBOEK	NAAM	ADRES	TELEFOONNUMMER		
WOORDENBOEK	WOORD	DEFINITIE			
CATALOGUS	ARTIKEL	BESCHRIJVING	GEWICHT	MAAT	
	PRIJS		BESTELNUMMER		
KOERSLIJST	AANDEEL	OMZET	HOOGSTE KOERS	LAAGSTE KOERS	

Figuur 1-1 Voorbeelden van kolomopschriften bij een paar gangbare databases op papier

U begrijpt nu wel zo ongeveer, wat een database voorstelt, en misschien vraagt u zich nu wel af: 'Wat is dan een *computer* database? Wat kan ik daarmee doen, wat ik zonder niet kan?' Een computerdatabase kan niets wat u zelf niet zou kunnen doen met een database op papier. Sommige dingen zijn echter in de praktijk niet haalbaar zonder computer. We hebben bijvoorbeeld allemaal wel eens een blaadje papier gevonden met een telefoonnummer erop – geen naam, alleen een nummer. Als we willen uitzoeken, wie dat nummer heeft, dan hebben we niet zoveel aan het telefoonboek. Als het telefoonboek echter de vorm van een computerdatabase zou hebben, kunnen we de computer vragen aan wie het telefoonnummer toebehoort, en onmiddellijk zal de naam verschijnen.

Nog een voorbeeld: stel dat u het telefoonnummer wilt hebben van iemand die Jansen heet en die woont aan het Damrak in Amsterdam. U kunt het Amsterdamse telefoonboek van de computer ('AMTELEF' zou dat kunnen heten) dan vragen om enkel de namen, adressen en telefoonnummers van alle Jansens op het Damrak. Misschien krijgt u nog wel meer dan één naam, maar in ieder geval minder namen dan bij alle Jansens in heel Amsterdam.

De computer lost natuurlijk niet alles op. Hij kan niets doen, waartoe u zelf niet in staat bent. Maar – en dat is de maar waar alles om draait – hij kan u *helpen* de gewenste dingen snel en gemakkelijk te doen. De computer is een stuk gereedschap dat u helpt dingen voor elkaar te krijgen, die anders gewoon niet op een handige manier voor elkaar te krijgen zijn.

Als u een eigen telefoonlijst wilt gebruiken, dan kan een eenvoudige database er ongeveer uitzien als afgebeeld in Figuur 1-2.

Naam	Adres	Telefoonnummer
Broeks, Joost	Fagelstr 9, Arnhem	085-511967
Broeks, Jopie	Karstr 36, Bemmel	08811-1253
Drent, Carla	Damstr 13, Lobith	08365-1255
Haafs, Wim	Raalt 25, Arnhem	085-897697
Hul, Joost	Rijndijk 3, Driel	08306-42078
Meloen, Joep	Kuilsmaat 1/317, Zevenaar	08360-21253
Roos, Sylvia	De Eik 105, Didam	08362-3436
Joosten, Ina	Laantje 5, Ressen	08811-4027

Figuur 1-2 Een voorbeelddatabase

Een echte database kan natuurlijk veel en veel meer regels informatie bevatten. De kleine database hierboven kunt u beter in een klein opschrijfboekje bewaren dan in een computer. Dan kunt u hem meenemen, er aantekeningen in maken, en het is nog goedkoper ook. Wilt u waar voor uw computergeld, dan hebt u een heleboel informatie nodig – in de regel zoveel, dat u die informatie zonder computer niet doelmatig kunt gebruiken. Het doel van dit boek is u te vertellen over databases en hoe u ze gebruikt. Daarom gaan we heel kleine databases gebruiken als voorbeelden. Precies dezelfde beginselen zouden van toepassing zijn, als deze persoonlijke telefoonlijst het telefoonboek was voor het district Arnhem. Er bestaat geen enkel verschil, behalve dan dat het telefoonboek heel wat meer vermeldingen omvat.

Wij willen u meer vertrouwd maken met computerdatabases, namelijk:

- het plan opstellen
- het maken
- het gebruik en
- het veranderen ervan

Daarvoor gaan we een computerdatabase maken van het eenvoudige telefoonlijstvoorbeeld. Daarbij gaan we het databasesysteem 'dBASE II' voor microcomputers gebruiken. dBASE II is representatief voor de betere database management systemen die op het moment beschikbaar zijn voor microcomputers. Een database management systeem is een programma dat op een computer geïnstalleerd kan worden, alle details van de database voor zijn rekening neemt en de gebruiker in staat stelt gebruik te maken van de inhoud van de database, die te bewerken en te veranderen.

6 ... DATABASES

Als u een microcomputer bezit met dBASE II of toegang ertoe hebt, kunt u de aanwijzingen in dit boek uitvoeren en met gebruikmaking van uw computer met het boek gelijkop werken. Als u geen microcomputer hebt, of geen dBASE II, kunt u toch zonder problemen doorgaan. Bij elke stap zal een beschrijving gegeven worden van wat u doet en van wat de computer doet. Wat de computer 'doet', komt in de schermafbeeldingen in gewoon lettertype te staan; wat u 'doet' – wat u INTYPT – komt in VET lettertype te staan.

VAN PERSOONLIJKE TELEFOONLIJST TOT COMPUTERDATABASE: DE TERMINOLOGIE

Laten we nogmaals kijken naar de gegevens uit ons eigen telefoonlijstje.

HET RECORD

Broeks, Joost Fagelstr 9, Arnhem 085-511967

De regel hierboven heet een RECORD. De stukken informatie die samen een RECORD vormen staan horizontaal – weergegeven in rijen op het scherm. Ons telefoonlijstje heeft acht rijen – acht RECORDS – acht keer naam + adres + telefoonnummer.

HET VELD

Als we verticale lijnen zouden trekken tussen de namen en adressen en tussen de adressen en de telefoonnummers in onze telefoonlijst, zouden we kolommen isoleren met soortgelijke informatie. We zouden drie afzonderlijke groepen verticaal gerangschikte gegevens van elkaar scheiden – een kolom met namen, een kolom met adressen en een kolom met telefoonnummers. In computerterminologie heten deze kolommen VELDEN.

VELDNAMEN

De opschriften van de kolommen, NAAM, ADRES en TELEFOONNUMMER, heten VELDNAMEN.

Een computerdatabase is in principe hetzelfde als eentje die u met potlood en papier zou kunnen maken. Het moge duidelijk zijn, dat zowel een database op papier als een database in een computer er zijn om GEBRUIKT te worden. Het is dus goed om te vragen: wat DOET u eigenlijk met uw telefoonlijstje?

U schrijft nieuwe kennissen in, verandert misschien namen, adressen of telefoonnummers van mensen, die trouwen of gaan scheiden, verhuizen, of een nieuw telefoonnummer krijgen. Misschien streept u bepaalde mensen door (of gumt u ze uit, als u zo'n vooruitziende blik hebt gehad, dat u uw records met potlood hebt ingevuld ...). Wanneer u de hier opgeslagen informatie wilt gebruiken, dan zal het wel zo zijn, dat u iemand wilt proberen op te bellen, dat u naar een feestje op een bepaald adres wilt, of dat u een brief wilt versturen.

Uw telefoonlijstje weerspiegelt een veranderingsproces – als u het goed bijhoudt – en zal u op elk willekeurig moment de informatie leveren die u zoekt. Hetzelfde geldt voor uw computerdatabase.

U kunt heel gemakkelijk informatie aan een computerdatabase toevoegen, informatie eruit weghalen of de informatie erin veranderen. Evenzo kunt u informatie uit uw computerdatabase lichten om te bekijken – en nog heel gemakkelijk ook.

Bij wat u zo dagelijks doet, bent u de hele tijd bezig dingen toe te voegen aan de aanwezige informatie, of juist dingen eruit weg te nemen, te wijzigen, te lezen wat u wilt zien, en te negeren wat u niet wilt zien. Deze activiteiten vormen een wezenlijk bestanddeel van onze intelligentie – wat er gebeurt, is echt iets heel gewoons.

We gaan nu echter al deze informatie, die we zo gewend zijn overal vrijelijk om ons heen te vinden, in een 'computerdatabase' stoppen. Er zal bij wijze van spreken een informatiedrager tussen ons en de informatie in komen te staan. We zullen eraan moeten wennen dat deze er is en dat we de informatie tevoorschijn kunnen halen, wanneer we daaraan behoefte hebben. Als u eenmaal aan de nieuwe situatie gewend bent, zult u verbaasd staan, hoeveel een computerdatabase voor u kan doen.

Het gebruik van uw computer zal net zo gemakkelijk worden als het gebruik van uw telefoonlijstje. Net als alles wat u doet in dit leven, moet u er een klein beetje bij nadenken, heeft u een klein beetje planning nodig en een beetje een gebruiksaanwijzing. U moet weten hoe u de informatievoorraad moet aanleggen, gebruiken en veranderen. Bij dit leerproces hoeft u niet alles te begrijpen, u hoeft het wiel niet opnieuw uit te vinden. U leert enkel een nieuwe voorziening te beheersen – een nieuw mechaniek voor een nieuwe machine – die is ontworpen om uw pogingen te ondersteunen al die soorten informatie waarmee u al door en door vertrouwd bent, te hanteren en te bewerken.

U krijgt nu dus computerdatabases te zien. U gaat eenvoudige opbouwen en gebruiken. Om ervoor te zorgen, dat u de beginselen en oefeningen gemakkelijk in de praktijk kunt brengen, gebruiken we dBASE II. dBASE II is representatief voor de database management systemen die op het moment beschikbaar zijn voor microcomputers.

EEN PAAR EENVOUDIGE ANALOGIEËN

Laten we nog eens een ander voorbeeld bekijken om het idee achter een database management systeem te verduidelijken. Stel dat u een onderdeel voor uw auto nodig hebt. U kunt naar een auto-onderdelenwinkel toegaan en de verkoper vertellen, welk onderdeel u wenst. Hij zoekt dat onderdeel op in een stel onderdelencatalogi.

- Het eerste boek geeft hem het nummer van dat onderdeel.
- Hij zoekt dan dat onderdeelnummer op in een ander boek. Dit boek vertelt hem, waar in het magazijn dat onderdeel zich bevindt.
- Nadat hij het onderdeel heeft gehaald, gebruikt hij opnieuw het nummer om de prijs van het onderdeel in een prijslijst op te zoeken.

In dit analoge geval corresponderen de feitelijke auto-onderdelen met de gegevens in de database. De verkoper, de catalogi, de lijsten, de magazijnstellingen enzovoorts corresponderen met het database management systeem. Om gebruik te kunnen maken van dit 'auto-onderdelen management systeem' vertelt u de verkoper wat u wenst, in een taal die hij begrijpt (Nederlands), waarbij u de terminologie gebruikt die hoort bij auto's. 'Ik heb een carburateur nodig voor een Volkswagen Golf uit 1976.' De verkoper regelt verder alles: de onderdelen halen, de boeken bijhouden, weten hoe hij die boeken moet gebruiken, enzovoorts. Het enige wat u hoeft te doen, is enig idee hebben van wat u wenst. De verkoper en zijn boeken en catalogi zorgen voor de rest.

Hetzelfde geldt voor het database management systeem (DBMS) van een computer. Zodra een DBMS op een computer is 'geïnstalleerd', wordt de computer een 'expert' op het gebied van alle details die komen kijken bij het opbergen, catalogiseren en opzoeken van gegevens. Alles wat u hoeft te doen, is enig idee hebben van wat u wenst en een beetje computerterminologie kennen. Dit boek verschaft u de benodigde computerterminologie. De computer en het database management systeem zorgen voor de rest.

U hoeft u overigens niet te laten afschrikken door de eis dat het database management systeem op de computer 'geïnstalleerd' moet worden. Het lijkt in de verste verte niet op het speelgoed dat kinderen krijgen zogenaamd om even zelf in elkaar te zetten. Het is echt eenvoudig.

U kunt een database management systeem ook vergelijken met een grote bibliotheek. In veel grote bibliotheken, vooral in universiteitsbibliotheken, mag de gebruiker meestal niet bij de rekken komen, waar de boeken staan. Om een boek te kunnen krijgen moet u de catalogus raadplegen, informatie van het cataloguskaartje op een bonnetje schrijven en dat bonnetje aan een bibliothecaresse afgeven.

Vervolgens zal een 'kabouter', voor u onzichtbaar, door donkere gewelven ijlen om het boek op te halen en door te geven aan de bibliothecaresse, die het aan u overhandigt.

Wanneer u het boek teruggeeft, gebeurt zo ongeveer het omgekeerde van dit proces. De bibliothecaresse speelt het boek door aan een 'kabouter', die zich zal haasten het boek terug te zetten op de plaats waar het oorspronkelijk stond. Ook in dit geval corresponderen de catalogus, de bibliothecaresse, de kabouters en de gewelven met een database management systeem in een computer. Het boek correspondeert met het gegeven. Alles dat van u gevraagd wordt, is een beetje kennis, hoe u dit systeem moet gebruiken. Het systeem doet zelf al het werk.

ONZE EERSTE COMPUTERDATABASE

We hebben het uitvoerig gehad over ons persoonlijke telefoonlijstje. Dat lijstje is een goed uitgangspunt voor onze eerste computerdatabase. Wij zullen het gebruiken als voorbeeld en al doende zult u zien dat wat we doen, niet zo heel erg veel verschilt van het typen op een gewoon vel papier met een gewone schrijfmachine.

Stel dat we deze informatie op een vel papier zouden gaan typen. Degene die dit intikt, zou misschien het volgende doen:

1 de kolomopschriften invoeren, dus Naam, Adres en Telefoonnummer;

2 uitzoeken hoeveel posities voor iedere kolom gebruikt moeten worden om de zaak netjes en ordelijk te houden; en

3 een bladzijde-opschrift of titel typen.

Bij een computerdatabase moeten deze drie dingen ook echt altijd alle drie gedaan worden.

- U moet VELDNAMEN geven aan iedere kolom (ieder VELD).
- U moet de breedte voor elke kolom uitzoeken – dit heet de VELDBREEDTE.
- En u moet de database een naam geven. Het opschrift van de database heet de BESTANDSNAAM.

We zijn nu bijna zover, dat we een gecomputeriseerde versie kunnen gaan maken van uw persoonlijke telefoonlijstje. Maar voordat we verder gaan, moet u weten, dat computers vreemde beperkingen kennen.

Als u potlood en papier ter hand neemt om een lijstje te maken van de namen, adressen en telefoonnummers van uw vrienden, dan kunt u de lijst achteraf een opschrift geven, bijvoorbeeld 'telefoonklapper' of 'adreslijstje'. Bij de computer is het omgekeerd. U *moet* eerst een titel geven, voordat hij de lijst accepteert.

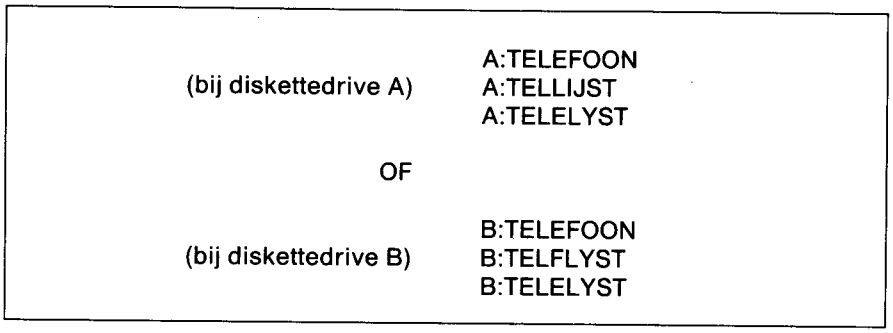
In het algemeen heet een database 'file' en de titel heet FILENAAM. FILENAMEN hebben bepaalde bijzondere eigenschappen.

- Ze kunnen niet uit meer dan ACHT letters en cijfers bestaan.
- Er mogen geen open plaatsen of spaties in staan.
- Ze moeten met een LETTER beginnen.

Verder wordt de filenaam gewoonlijk voorafgegaan door een extra symbool. Dit vertelt de computer, welke diskettedrive voor de gegevensfile gebruikt moet worden. Een diskettedrive is een apparaat dat vaak gebruikt wordt om informatie op te bergen. Opslagapparaten bij computers, zoals diskettedrives, zullen uitgebreider besproken worden in Hoofdstuk 3.

Vaak is er meer dan één dergelijke diskettedrive aangesloten op de computer. Het kenmerk voor de diskettedrive is de methode om de computer ervan op de hoogte te stellen, welk apparaat gebruikt moet worden. In deze tekst worden de diskettedrives aangegeven met een letter gevolgd door een dubbele punt – bijvoorbeeld 'A:'. Dit is een conventie die door vele microcomputersystemen gebruikt wordt.

In Figuur 1-3 staan een paar mogelijke titels (FILENAMEN) voor ons gecomputeriseerde telefoonlijstje (onze database). Bedenkt u, dat u de database elke willekeurige naam van 8 of minder letters kunt geven.



Figuur 1-3 Mogelijke databasenamen

De filenamen in bovenstaand voorbeeld heten mnemonics. Een mnemonic is geen echt woord, maar een groep letters, waarvan de uitspraak een aanwijzing geeft over de betekenis ervan. Het is nog niet eens zo'n gek idee filenamen op deze manier te kiezen. De volgende vijf 'woorden' zijn voorbeelden van mnemonics die in aanmerking komen als filenamen.

CREDITRN DEBITERN LOONLYST PERSONEL KWRTLBTW

VELDNAAM

Elke kolom in de database (elk VELD) moet een kolomopschrift krijgen (een VELDNAAM). Voor VELDNAMEN bestaan net zulke beperkingen als voor FILENAMEN.

- Ze kunnen niet uit meer dan TIEN letters en cijfers bestaan.
- Er mogen geen spaties in staan.
- Ze moeten met een LETTER beginnen.

Onze eenvoudige telefoonlijst heeft drie kolomopschriften (VELDNAMEN); N A A M, A D R E S en T E L E F O O N N U M M E R. De eerste twee (NAAM, ADRES) zijn als VELDNAMEN bruikbaar, TELEFOONNUMMER echter niet. Dat woord heeft meer dan tien letters. We moeten deze kolom dus voorzien van een mnemonic naam, zoals TELFNUM of TELEFOON of NUMMER, om te voldoen aan de 10-teken-regel.

VELDTYPE

Het is ook nodig de computer te vertellen, hoeveel tekens (letters, cijfers, spaties en andere symbolen) voor elk veld nodig zijn. U gaat een getal invoeren, dat gelijk is aan het aantal 'vakjes', dat nodig is om de tekens, spaties en andere symbolen te bevatten.

Omdat de computer anders omgaat met verschillende soorten velden, is het nodig te weten, welk van de drie beschikbare velden in onze behoeften voorziet. Velden zijn er in drie soorten:

met TEKENS (characters)

met GETALLEN (numeriek), of

LOGISCH

In ons voorbeeld zijn alle VELDEN TEKENVELDEN. Ze kunnen letters, cijfers, spaties en andere standaard schrijfmachinesymbolen bevatten. In toekomstige voorbeelden zullen LOGISCHE en GETALVELDEN nader worden verklaard.

Er zijn een paar dingen, die de computer uit zichzelf doet. Telkens wanneer een record wordt ingevoerd, kent de computer er automatisch een nummer aan toe. De computer noemt de eerste rij, dus het eerste record, RECORD 1, het tweede record RECORD 2 enzovoorts. Dat is aardig, maar misschien denkt u: 'Nou en?' U zou inderdaad lange tijd met databases kunnen werken en behoorlijk ingewikkeld werk ermee kunnen doen zonder ooit een recordnummer te gebruiken. Voor bepaalde dingen zijn die recordnummers echter handig en het nut ervan zullen we verderop in dit boek bespreken.

We hebben nu wat van de belangrijkste terminologie besproken. Zorgt u ervoor, dat u goed vertrouwd bent met deze begrippen, omdat we als volgende stap een eenvoudige database gaan opzetten. Denkt u eraan dat:

Rijen records heten en automatisch door de computer van een nummer worden voorzien;

Kolommen velden heten: ze hebben opschriften nodig, VELDNAMEN genaamd (maximaal 10 tekens, geen spaties, beginnend met een letter);

U uw database een titel moet geven, met andere woorden een FILENAAM (maximaal 8 tekens, geen spaties, beginnend met een letter);

U de computer moet vertellen, waar (in welke diskettedrive) de gegevens weggezet moeten worden (Voorbeeld: A:TELEFOON zal de gegevens in drive 'A' zetten);

U de computer de vorm en de indeling van de informatie moet opgeven, die u gaat invoeren. U bent bekend met FILENAAM en VELDNAMEN. Maak de taak af door (1) het aantal posities toe te wijzen nodig om plaats te bieden aan de informatie voor de verschillende velden en (2) de computer te vertellen of een veld het type tekens (characters), numeriek of logisch heeft.

ONZE EERSTE OEFENING

In deze eerste oefening gaan we werkelijk bij het begin beginnen. Wanneer u de computer net aanzet, zal het scherm er ongeveer uit komen te zien als Scherm 1-1.

```
*CP/M 2.2

A>
```

Scherm 1-1

HET BESTURINGSSYSTEEM

*CP/M 2.2 is een veelgebruikt 'besturingssysteem' voor microcomputers. Een besturingssysteem helpt u bij het bedienen van de computer. *CP/M staat voor Control Program for Microcomputers. Veel database management systemen, die op de markt zijn, zoals dBASE II, gebruiken het besturingssysteem om routinehandelingen te verrichten. U zult u niet hoeven te verdiepen in de details ervan om een database management systeem te kunnen gebruiken.

Het besturingssysteem legt soms bepaalde manieren om dingen te doen op aan het database systeem. In alle voorbeelden in dit boek zullen we de conventies van het *CP/M besturingssysteem gebruiken. *CP/M is een product van Digital Research in Pacific Grove (Californië). Het nummer 2.2 is het versienummer. Een versienummer is net zoiets als een typenummer.

*CP/M is een gedeponeerd handelsmerk van Digital Research, Inc.

De A> is een PROMPT. Het is de manier waarop het besturingssysteem u vertelt 'Ik sta klaar. Vertel me wat ik moet doen.' Het symbool rechts van de > heet de cursor. De cursor is een knipperend streepje, dat op het scherm verschijnt om u duidelijk te maken, waar u bent. Voor de computer is de cursor zo ongeveer hetzelfde als de punt van een potlood. Letters die u op het scherm 'typt' door toetsen aan te slaan op het toetsenbord, zullen daar verschijnen, waar de cursor zich bevindt. Na elke getypte letter verschuift de cursor naar rechts.

UW COMPUTER GAAT dBASE II GEBRUIKEN

De computer gereed maken voor het gebruik van uw database management systeem heet het LADEN van de database. Om te beginnen typt u gewoon de letters DBASE in. Het beeldscherm zal er nu uitzien als Scherm 1-2.

```
CP/M 2.2

A> DBASE◆  (Opmerking: de cursor wordt aangegeven met een ruitje)
```

Scherm 1-2

Druk op de RETURN toets. De RETURN is een heel interessante toets. De naam is ontleend aan de 'wagen-terug' van een electrische schrijfmachine. Bij computerwerk zou deze toets eigenlijk INVOER toets of iets dergelijks moeten heten. Bij de meeste bewerkingen geeft u door het indrukken van deze toets aan de computer te kennen, dat u klaar bent met het intypen van een regel: 'Voer die regel nu maar uit.'

De computer zal antwoorden als afgebeeld in Scherm 1-3.

```
CP/M 2.2

A> DBASE

PLEASE ENTER TODAY'S DATE DD/MM/YY OR PRESS RETURN:◆
```

Scherm 1-3

De cursor staat nu te wachten totdat u de datum in de gevraagde vorm invoert. Doet u dit nu. Het scherm komt er nu uit te zien als afgebeeld in Scherm 1-4. Merkt u op, dat het beeld bovenaan het scherm is begonnen en zich nu naar beneden toe heeft uitgebreid.

```
CP/M 2.2

A > DBASE

PLEASE ENTER TODAY'S DATE DD/MM/YY OR PRESS RETURN: 29/1/85

***DBASE II          VERSION 2.43          15 FEB 1982
◆
```

Scherm 1-4

De punt helemaal links onderaan het scherm onder de sterretjes is van groot belang. In dBASE II wordt deze prompt de PUNTPROMPT genoemd. De PUNTPROMPT is de manier waarop de computer u vertelt 'Ik ben klaar, geef me opdracht iets te doen.' De dingen die u de computer kunt opdragen heten COMMANDO's.

Wilt u dBASE II verlaten, dan typt u eenvoudigweg het woord QUIT achter een puntprompt:

.QUIT

De computer zou dan antwoorden met:

```
*** End run   dBASE II  ***

A>
```

U bent dan terug bij het besturingssysteem van uw computer. Maar dit doet u nu niet; we gaan verder vanaf Scherm 1-4.

De regel onder het verzoek om de datum laat u weten, wat u nu precies hebt 'geïnstalleerd'. dBASE II is de naam van het database management systeem. 'Versie 2.43' laat u weten, welke 'druk' van dBASE II u gebruikt en de datum daarachter is de verschijningsdatum van versie 2.43. Een versienummer is net zoiets als een typenummer. Gewoonlijk heeft iedere nieuwe versie alle voorzieningen van de voorafgaande versies plus nieuwe en/of verbeterde voorzieningen.

We zijn nu zover, dat we kunnen beginnen met het MAKEN van de database B:TELEFOON. Het gesprek tussen de gebruiker en de computer wordt weergegeven in Scherm 1-5. De antwoorden van de computer (de PROMPTS) en de overeenkomstige toetsinvoer zijn weergegeven zoals ze eruit zouden kunnen zien op een beeldscherm. De prompts van de computer worden weergegeven in normaal lettertype. De invoer via het toetsenbord is hier **vetgezet**. Op een beeldscherm zou de toetsenbordinvoer in werkelijkheid de normale intensiteit hebben, terwijl de aansporingen van de computer in gereduceerde intensiteit zouden verschijnen. Merkt u op, dat hetgeen we tot nu toe gedaan hebben, op het scherm is blijven staan, terwijl we doorwerkten...

```
CP/M 2.2

A> DBASE

PLEASE ENTER TODAY'S DATE DD/MM/YY OR PRESS RETURN: 3/1/85

***DBASE II VERSION 2.3 15 FEB 1982

.CREATE
ENTER FILENAME: B:TELEFOON
ENTER RECORD STRUCTURE AS FOLLOWS:
FIELD NAME, TYPE, WIDTH, DECIMALPLACES
001 NAAM, C, 15
002 ADRES, C, 25
003 TELEFOON, C, 12
004

INPUT DATA NOW? Y ♦
```

Scherm 1-5

We gaan nu de informatie invoeren, die voor de computer de vorm en de indeling van de database aangeeft.

Meteen achter de PUNTPROMPT typt u het woord CREATE in. Vervolgens drukt u op de RETURN toets. CREATE is het dBASE II COMMANDO dat het begin aangeeft van het opzetten van een database. RETURN laat de computer het COMMANDO accepteren, dat u net hebt ingetypt (CREATE). Denkt u eraan, dat de computer niets doet, voordat u RETURN aanslaat.

Nadat u CREATE hebt ingevoerd (en op de RETURN toets hebt gedrukt), zal de computer antwoorden met de prompt ENTER FILENAME:. De juiste invoer via het toetsenbord is – diskettedrive-aanduiding, dubbele punt en de filenaam van acht letters (en cijfers). In ons voorbeeld is via het toetsenbord B:TELEFOON ingevoerd.

De volgende aansporing van de computer zal u vragen de veldnaam in te voeren, het veldtype en de afmeting van het veld (hoeveel tekens er zijn). Er verschijnt

```
ENTER RECORD STRUCTURE AS FOLLOWS:
FIELD          NAME, TYPE, WIDTH, DECIMAL PLACES
001
```

Denkt u erom, dat de veldnaam uit maximaal tien letters en cijfers kan bestaan, te beginnen met een LETTER. Een veldnaam mag geen spatie bevatten.

Het eerste veld is een tekenveld waarvan de VELDNAAM NAAM is. We besluiten 15 tekens voor dit veld te gebruiken. De invoer op het toetsenbord is **NAAM, C, 15.** Druk vervolgens op de RETURN toets. U ziet dat de prompt van de computer ook om decimale posities vraagt. Daarvoor werd niets op het toetsenbord ingevoerd en dat was ook niet nodig, omdat het veld een tekenveld is. Voor tekenvelden negeren we het verzoek van de computer om decimale posities. De hoofdletter C tussen de twee komma's maakt de computer duidelijk, dat dit een teken- (character-) veld is. Er zijn komma's nodig tussen veldnaam, veldtype, breedte en (zo die worden opgegeven) decimale posities.

De computer liet ook een veldnummer 001 zien in zijn aansporing. Dit gaat automatisch, net als het nummeren van records door de computer. Dit betekent dat de informatie die u gaat invoeren, de beschrijving geeft van het eerste veld.

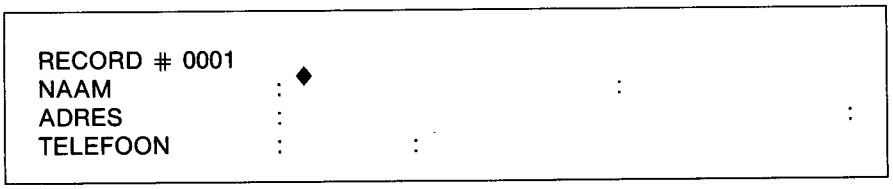
Wanneer u de RETURN toets indrukt na **NAAM, C, 15** antwoordt de computer onmiddellijk met 002. Dit vertelt u dat de computer nu gereed is om de veldnaam, het veldtype en de breedte van het tweede veld op te nemen. Dit zal net zolang doorgaan, totdat u ofwel twee-en-dertig veldbeschrijvingen hebt ingevoerd (het maximum voor dBASE II) danwel een einde hebt gemaakt aan het invoeren. Maakt u zich niet bezorgd, als u denkt, dat u ooit meer dan 32 veldnamen nodig zult hebben – dat kan, maar het is wat ingewikkelder. Later zult u zien, hoe u kunt zorgen voor meer dan 32 velden, als uw database er zoveel nodig heeft.

Het invoeren kan ieder willekeurig moment worden afgebroken door op de RETURN toets te drukken in plaats van de velddefinitie in te typen. In het voorbeeld werd RETURN ingedrukt, toen de computer vroeg om de definitie van veld 004. Dat deden we, omdat we maar drie velden gebruiken in deze database (NAAM, ADRES, en TELEFOON).

GEGEVENS INVOEREN

De computer zal u nu vragen, of u nu gegevens wilt invoeren. U moet nu met de Y van yes antwoorden voor ja of met de N van nee. U hoeft nu niet op de RETURN toets te drukken; de computer zal in dit geval onmiddellijk gehoor geven aan de Y of N.

Een bevestigend antwoord zal tot gevolg hebben, dat de computer het beeldscherm wist en u begint aan te sporen de namen, adressen en telefoonnummers (de gegevens) in te voeren. Voert u 'Y' in en uw beeldscherm zal eruit komen te zien als afgebeeld in Scherm 1-6.



Scherm 1-6

De tekst die de computer heeft geproduceerd, zal verschijnen met gereduceerde intensiteit. De dubbele punten geven het begin en het einde aan van de ruimte die is toegewezen aan het veld waarvan de veldnaam links van de meest linkse dubbele punt staat.

Dit soort weergave is een schermvullende prompt. Het scherm spoort u aan alle informatie voor het hele scherm te verschaffen (in tegenstelling tot een puntprompt, die enkel om één regel invoer vraagt). De computer toont de veldnamen en een ruimte waarvan verwacht wordt, dat u daarin de informatie invoert, die bij het veld (bij de kolom) behoort.

Bij het invoeren van de gegevens voor het naamveld bijvoorbeeld zal de cursor zich naar rechts verplaatsen, totdat het veld vol is. Er zal dan een biepje weerklinken en de cursor zal naar de meest linkse positie van het volgende veld springen. Het meest waarschijnlijk is echter, dat de ingevoerde naam de ruimte toegewezen aan het naamveld niet geheel vult. U moet dan op de RETURN toets drukken, wanneer de naam volledig is ingevoerd. Wanneer u op de RETURN toets drukt, zal de cursor zich verplaatsen naar het volgende veld (in dit geval het ADRESveld).

Wanneer u alle gegevens van een bepaald record hebt ingevoerd, als getoond in Scherm 1-7, zal de computer automatisch het scherm schoonmaken en beginnen u aan te sporen gegevens in te voeren voor het volgende record. Dit gaat door totdat u alle gegevens hebt ingevoerd.

```
RECORD # 00001

NAAM        :Broeks, Joost          :
ADRES       :Fagelstr 9, Arnhem     :
TELEFOON    :085-511967:
```

Scherm 1-7

Velden hadden we hiervoor *verticale* kolommen genoemd. Nu worden de gegevens voor de verticale kolommen ingevoerd in de vorm van opeenvolgende rijen op het scherm. Dit wordt gedaan voor het gemak bij het invoeren van gegevens. De computer zet elk ingevoerd veld in een 'verticale kolom' in de database. Wanneer u de gegevens voor de acht records van ons voorbeeld hebt ingevoerd, dan zal het scherm eruit zien als Scherm 1-6, behalve dat het recordnummer 00009 is; de computer wacht op het negende record. Maar we hebben ditmaal geen negende.

Als u nu op de RETURN toets drukt – met de cursor op de getoonde plaats – zal de computer CREATE verlaten en antwoorden met een PUNT-PROMPT, waarmee hij aangeeft, dat hij gereed is verdere opdrachten in ontvangst te nemen.

DATABASES GEBRUIKEN

Nu hebben we een database. Hij heeft acht records en drie velden; NAAM, ADRES, en TELEFOON. De FILENAAM (titel) ervan is TELEFOON en hij bevindt zich in de B: drive. Als we de database willen gebruiken, typen we USE B:TELEFOON (en een RETURN) achter een PUNT-PROMPT.

```
.USE B:TELEFOON
```

De computer antwoord met nog een PUNTPROMPT op de volgende regel. Misschien hebt u wel een heleboel databases op uw computer beschikbaar. Via 'B:TELEFOON' vertelt u de computer dat u met die database wilt werken, die zich in de B: drive bevindt, waarvan de FILENAAM TELEFOON is. USE is het dBASE II COMMANDO, dat zo ongeveer hetzelfde is als tegen uw assistent zeggen: 'James, geef me de telefoonlijst eens aan.' De transactie zal er zo uitzien:

```
.USE B:TELEFOON
.◆
```

Wanneer de computer antwoordt met de PUNTPROMPT op de regel net onder .USE B:TELEFOON, is de database TELEFOON klaar voor gebruik.

Laten we het commando eerst gebruiken om te bekijken, wat we net opgebouwd hebben. De opbouw van de database wordt in feite bepaald door de definitie van de VELDEN (kolommen). Om die opbouw nog eens te kunnen bekijken typt u DISPLAY STRUCTURE achter de PUNTPROMPT. De computer zal antwoorden met het schermbeeld afgebeeld in Scherm 1-8.

```
.DISPLAY STRUCTURE                          (toetsenbordcommando)

STRUCTURE FOR FILE:      B:TELEFOON.DBF
NUMBER OF RECORDS:       00008
DATE OF LAST UPDATE:     9/15/81
PRIMARY USE DATABASE
FLD     NAME       TYPE  WIDTH   DEC   (antwoord van computer)

001     NAAM       C      015
002     ADRES      C      025
003     TELEFOON   C      012

  **TOTAL**               00053
```

Scherm 1-8

Merkt u op, dat de computer .DBF heeft toegevoegd aan de FILENAAM van de database. Merkt u ook op, dat het "TOTAL" het totale aantal zichtbare tekens plus één is, dat in één record van de database gebruikt is.

.DBF is een 'filetype'. Voor andere files, die we later zullen tegenkomen, zal de computer ook een filetype toevoegen. Het filetype 'vertelt' de computer, hoe de file behandeld moet worden. De datum van de laatste wijziging is de datum die 'ingetoetst' werd, toen we dBASE II binnengingen. Iedere keer dat de database wordt 'binnengegaan', wordt een datum ingetoetst. Als iets in de database wordt veranderd, zal de datum van de laatste wijziging veranderd worden. Als u enkel RETURN aanslaat – het alternatief voor het invoeren van een datum – komt de datum van de laatste wijziging eruit te zien als 00/00/00. De regel 'totaal', die onderaan het scherm staat, geeft het aantal tekens (plus 1) aan in een record. Dit is de aanduiding voor de recordgrootte, waarvan het belang later besproken zal worden.

Nu we B:TELEFOON hebben 'gebruikt' om te bekijken, wat voor structuur we gemaakt hadden, kunnen we overgaan tot het 'hoofddoel' van databases: het opslaan en weergeven van informatie. Stel dat we wat van de informatie willen terughalen, die we opgeslagen hebben, om ernaar te kijken. Eén manier om naar de informatie te kijken is het typen van DISPLAY ALL achter een PUNTPROMPT. DISPLAY ALL is het dBASE II commando dat de inhoud van de database weergeeft. De uitkomst van dit commando is te zien in Scherm 1-9.

```
.USE B:TELEFOON
.DISPLAY ALL

00001  Broeks, Joost   Fagelstr 9, Arnhem        085-511967
00002  Broeks, Jopie   Karstr 36, Bemmel        08811-1253
00003  Drent, Carla    Damstr 13, Lobith        08365-1255
00004  Haafs, Wim      Raalt 25, Arnhem         085-897697
00005  Hul, Joost      Rijndijk 3, Driel        08306-42078
00006  Meloen, Joep    Kuilsmaat 1/317, Zevenaar 08360-21253
00007  Roos, Sylvia    De Eik 105, Didam        08362-3436
00008  Joosten, Ina    Laantje 5, Ressen        08811-4027
```

Scherm 1-9

De linkerkolom toont het automatisch toegewezen RECORDNUMMER waar we het al eerder over gehad hebben. Als u het recordnummer niet weergegeven wilt hebben, typt u DISPLAY ALL OFF in plaats van DISPLAY ALL. De computer zal hetzelfde schermbeeld tonen, behalve dat de recordnummers niet te zien zullen zijn.

Eerder in dit hoofdstuk gaven we een paar voorbeelden van dingen die de computer gemakkelijk kan doen. Een van die voorbeelden betrof een telefoonnummer op een stukje papier. Het dBASE II commando DISPLAY zal ons de bezitter van dat telefoonnummer leveren (als het in de database zit). Om dit voor elkaar te krijgen moeten we de computer vertellen, wat we willen dat hij doet, op zo'n manier dat hij het kan begrijpen. De computer mag dan snel zijn, hij is niet bijzonder slim.

We gebruiken dus het dBASE II commando DISPLAY om ons de 'bezitter' te leveren van het telefoonnummer op het stukje papier dat we gevonden hebben. Die transactie ziet er zo uit:

```
.DISPLAY FOR TELEFOON = '08362-3436'

00007      Roos, Sylvia      De Eik 105, Didam      08362-3436
```

Dit ziet er heel eenvoudig uit en dat is het ook. Het woord TELEFOON is de VELDNAAM van het VELD (de kolom) die de telefoonnummers bevat. Wat u de computer in feite vertelt met de korte instructie hiervoor, is: doorzoek de hele database en laat alle records zien, die '08362-3436' in de telefoonnummerkolom hebben staan.

De enkele aanhalingstekens aan weerszijden van het telefoonnummer zijn belangrijk. Als die er niet stonden, zou de computer niet weten, wat te doen, behalve dan u te vertellen, dat u een fout had gemaakt.

Deze enkele aanhalingstekens heten SCHEIDINGSTEKENS. Ze staan er om de computer te wijzen op het begin en het einde van een zogenaamde 'TEKENKETEN'. Wat tussen de aanhalingstekens staat, is teken voor teken, letterlijk, hetgeen u zoekt.

Telefoonnummers zijn tekenketens omdat we bij het maken van de database hebben aangegeven dat het telefoonnummerveld (TELEFOON) een tekenveld was. Cijfers zijn of getallen of tekens. Door het gebruik van scheidingstekens moet u de computer duidelijk maken welk van beide. Voor de computer is 555 bijvoorbeeld een getal, terwijl '555' een tekenketen is. Getallen moeten doorgaans als tekens ingevoerd worden, tenzij ze gebruikt worden in berekeningen.

Als volgende voorbeeld gaan we iedereen opsporen, die in Arnhem woont. Deze transactie gaat als volgt:

```
.DISPLAY FOR 'Arnhem'$ADRES

00001      Broeks, Joost      Fagelstr 9, Arnhem      085-511967
00004      Haafs, Wim         Raalt 25, Arnhem        085-897697
```

Net als eerst is de tekenketen omgeven door enkele aanhalingstekens. Dit voorbeeld is echter een beetje minder alledaags. Het dollarteken is een soort steno voor 'vervat in'. In feite geven we de machine de volgende opdracht: doorzoek de hele database en toon alle records die de tekens 'Arnhem' bevatten in de adreskolom. Als er ergens een Arnhemseweg in dat veld van de database staat, wordt ook dat record gemeld samen met de twee Arnhemse 'stadsadressen'. Wat de machine ook in dit geval zoekt, is *een reeks tekens.* Om het idee van een reeks tekens nader toe te lichten, geven we de computer de volgende opdracht:

```
.DISPLAY FOR 'ARNHEM'$ADRES
.
```

De computer antwoordt met een PUNTPROMPT. Dit betekent, dat er geen records zijn met de tekenketen ARNHEM in het adresveld. De twee tekenketens Arnhem en ARNHEM zijn verschillend. De eerste gebruikt kleine letters, terwijl de laatste alleen hoofdletters gebruikt. In dit voorbeeld weten u en ik, dat er hetzelfde bedoeld wordt, maar de computer weet dat niet. De computer vat alles erg letterlijk op.

Nog een voorbeeld van de letterlijke interpretatie van de computer is het volgende. Laten we de computer opdracht geven alle records te tonen, die de TEKENKETEN 'Joost' bevatten. Het antwoord van de computer is te zien in Scherm 1-10.

```
.DISPLAY FOR'Joost'$NAAM

00001   Broeks, Joost   Fagelstr 9, Arnhem   085-511967
00005   Hul, Joost      Rijndijk 3, Driel    08306-42078
00008   Joosten, Ina    Laantje 5, Ressen    08811-4027
```

Scherm 1-10

Dit is een tekenend voorbeeld van een tekenketenopsporing, waar de computer meer informatie heeft gegeven, dan waarom we dachten gevraagd te hebben. Het gaat hier om het volgende: *de computer heeft precies gedaan, waarom we gevraagd hadden.* In dit geval wilden we iedereen op het scherm hebben, wiens voornaam Joost is. Dat hadden we voor elkaar kunnen krijgen door ', Joost' te gebruiken in plaats van gewoon 'Joost' in de opdracht. Dat werkt, omdat 'Joost' als voornaam altijd door een komma en een spatie wordt voorafgegaan.

Om nog wat meer weg te nemen van de geheimzinnigheid van de werking van een computerdatabase (zo die geheimzinnigheid al bestond) gaan we kijken, wat er gebeurt wanneer we zeggen .DISPLAY FOR 'Arnhem'$ ADRES

- De computer 'gaat' naar het begin van de database (Record 1) en doorzoekt het adresveld van record 1 om te kijken of daar de TEKENKETEN 'Arnhem' inzit.
- Als die erin zit, zal de computer het hele Record 1 weergeven op het beeldscherm.
- Als die er niet inzit, zal de computer van Record 1 niets laten zien.
- Als Record 1 helemaal doorzocht is en al dan niet op het beeldscherm is weergegeven, zal de computer het zoeken vervolgen met Record 2, op precies dezelfde manier.
- Dit record voor record doorzoeken van het ADRESVELD gaat door totdat alle records onderzocht zijn.

Laten we eens even recapituleren, wat we tot nu toe gedaan hebben. We hebben wat van de terminologie geleerd, we weten dat databases veel gemeen hebben met dingen die we iedere dag gebruiken en dat databases en computers niets kunnen doen, dat u niet ook zonder al kunt doen – als u ongelofelijk veel tijd hebt.

Tot nu toe zouden potlood en papier veel sneller geweest zijn, goedkoper ook en bovendien zou u niets over computerterminologie hebben hoeven leren. Bedenkt u echter dat deze lijst gemakkelijk 80, 800 of 8000 namen zou kunnen tellen. De voorbeelden in dit boek gaan nooit over meer dan 15 records (om papier te besparen en uw belangstelling niet al te zeer op de proef te stellen). Om wat deze computertechnologie voor u kan doen, op zijn juiste waarde te kunnen schatten, zou u deze voorbeelden moeten beschouwen als kleine uittreksels uit databases met honderden of duizenden records.

Het lijkt allemaal erg veel op het maken van een tabel met potlood en papier.

- U moet een plan maken voor uw layout, besluiten welke informatie u in welke kolommen zet en vaststellen, hoe breed elke kolom moet zijn.
- Vervolgens voert u alle informatie in.
- Pas als dat alles gedaan is, kunt u de tabel gebruiken voor het doel waarvoor hij bedoeld is. In dit hoofdstuk hebben we de database maar op één manier gebruikt – we hebben hem bekeken, dat wil zeggen geDISPLAYd.

Het maken van een tabel op papier is helemaal analoog aan wat er gebeurt bij een computerdatabase. Voor de database moet eerst een plan gemaakt worden. Er moeten gegevens ingevoerd worden. Dan pas, en niet eerder, kunt u hem gebruiken om iets mee te doen.

Het fijne van dit alles is, dat u het allemaal kunt doen, zonder iets te weten over hoe de computer van binnen werkt. Alles wat u hoeft te doen, is de aanwijzingen op te volgen. Het is net zoiets als het rijden in een auto met een automatische versnellingsbak in plaats van een handgeschakelde.

De taal die u gebruikt om met de computer te 'praten' is een HEEL beperkt soort Engels. Bijna alle databases die op de markt zijn, hebben een 'Engelse' woordenschat van minder dan honderd woorden. In tegenstelling tot niet-computer Engels zijn er geen uitzonderingen. Elk woord betekent hetzelfde als toen het de laatste keer gebruikt werd. Elk woord in de woordenschat van de computer heeft een goed omschreven betekenis, waarbij geen ruimte bestaat voor afwijkende interpretaties. De betekenis is EEN van de gebruikelijke betekenissen van het woord.

Tenslotte hoeft u zich nergens druk om te maken. Bij het leren kunt u de computer niet kapot maken en als u iets niet helemaal zeker weet – dan kunt u gewoon het een en ander proberen en kijken wat werkt. Het moeilijkste dat u te leren krijgt, is waarschijnlijk dat er eigenlijk niets moeilijks is aan het werken met de computer. Het zou heel gemakkelijk onder de knie te krijgen moeten zijn. U gebruikt maar een heel beperkte woordenschat en er zijn geen moeilijke handelingen nodig (zoals het gebruiken van de versnelling in een auto).

2. ENKELE EENVOUDIGE TOEPASSINGEN

Nu u vertrouwd bent met wat een database is, zijn we zover, dat we een paar eenvoudige voorbeelden van databasegebruik kunnen bespreken en de werking van databases daarmee verder kunnen toelichten. Wij hebben al één eenvoudig commando besproken: DISPLAY. Daarmee hebben we gekeken naar de opbouw en de inhoud van een heel eenvoudige database.

In dit hoofdstuk gaan we leren, hoe we de database moeten veranderen – zowel door het toevoegen als door het wissen van informatie – gaan we standaardrapporten maken, een inhoudsopgave van de database vervaardigen en de inhoud van een database sorteren. Ook gaan we CREATE gebruiken om een wat ingewikkelder database te maken dan B:TELEFOON uit Hoofdstuk 1.

We hebben dus twee eenvoudige manieren besproken om de database te gebruiken:

- een manier om de opbouw ervan te bekijken en
- manieren om de hele inhoud van de database te bekijken of bepaalde delen ervan.

De volgende belangrijke activiteit met een database, waarmee we ons bezig zullen houden, is het veranderen ervan. Het is goed te weten, dat wijzigen gemakkelijk gaat.

Wat betreft het voorbeeld van de persoonlijke telefoonlijst kunnen we verwachten dat het eenvoudig is namen, adressen en telefoonnummers te 'wissen' en nieuwe ervoor in de plaats te zetten.

Dit is ook zo. Dat gemak moet er om verschillende redenen zijn. Bij het invoeren van gegevens worden fouten gemaakt – het verkeerde adres en telefoonnummer worden ingevoerd bij een bepaalde naam, woorden worden verkeerd gespeld, cijfers worden vergeten enzovoorts.

Mensen verhuizen, telefoonnummers en straatnamen veranderen. Sommige mensen verdwijnen uit ons leven – we maken kennis met anderen. Als de wereld werkelijk statisch was en alles bij het oude zou blijven, zouden databases aanmerkelijk minder nuttig zijn. De wereld staat echter allesbehalve stil. De samenleving verandert steeds sneller en het nut van databases is dan ook groot.

VERANDERING

Er zijn drie soorten verandering, waarmee een computer kan werken:

- ten eerste het toevoegen of wissen van hele records
- ten tweede het veranderen van de inhoud van een record
- tenslotte het veranderen van de opbouw van de database door het toevoegen of wissen van hele velden (hele kolommen).

RECORDS WISSEN

Het wissen van een record gebeurt in twee stappen, als beveiliging tegen abusievelijk wissen. Bij dBASE II worden daarvoor twee commando's gebruikt: DELETE en PACK. Om de werkwijze te verduidelijken, gaan we het record van Carla Drent (RECORD 3) wissen. De transactie met de computer is afgebeeld in Scherm 2-1.

```
.DELETE FOR NAAM = 'Drent, Carla'
00001 DELETION(S)
.DISPLAY FOR *
00003       *Drent, Carla       Damstr 13, Lobith       08365-1255
.PACK
PACK COMPLETE 00007 RECORDS COPIED
```

Scherm 2-1

We geven de computer de opdracht 'wis het record dat Carla Drent bevat'. Het DELETE commando zet een merkje (*) voor het record dat we willen verwijderen. De computer vertelt ons [met (*)] dat hij één record ter verwijdering heeft gemerkt.

Op dit moment zit het record nog steeds in de database. Als de computer nog een Carla Drent had gevonden, zou het antwoord zijn geweest '00002 DELETIONS'; voor ons een aanwijzing, dat er iets onverwachts was gebeurd.

Het commando DISPLAY FOR geeft de computer opdracht de records te tonen, die gemerkt zijn om verwijderd te worden. De computer toont één enkel record, het record, dat we wilden verwijderen. Merk op dat het symbool * tussen het RECORDNUMMER en het RECORD in staat.

Het commando PACK neemt het eigenlijke verwijderen van het record voor zijn rekening. Hiermee worden alle records verwijderd, die door het DELETE commando gemarkeerd zijn, en tegelijk worden de overblijvende records opnieuw genummerd.

Als we nu de hele database op het scherm halen, is het record van Carla Drent verdwenen. Er zijn nu slechts 7 records en het record dat eerst nummer 4 was, is nu nummer 3 enzovoorts. De herziene database staat afgebeeld in Scherm 2-2.

00001	Broeks, Joost	Fagelstr 9, Arnhem	085-511967
00002	Broeks, Jopie	Karstr 36, Bemmel	08811-1253
00003	Haafs, Wim	Raalt 25, Arnhem	085-897697
00004	Hul, Joost	Rijndijk 3, Driel	08306-42078
00005	Meloen, Joep	Kuilsmaat 1/317, Zevenaar	08360-21253
00006	Roos, Sylvia	De Eik 105, Didam	08362-3436
00007	Joosten, Ina	Laantje 5, Ressen	08811-4027

Scherm 2-2

RECORDS TOEVOEGEN (APPEND)

Het toevoegen van een record aan een database is een eenvoudig éénstapsproces. Bij dBASE II is het commando voor het toevoegen van een record APPEND. Het invoeren van gegevens voor APPEND gaat precies zo als het invoeren van gegevens voor CREATE. Wanneer de computer opdracht krijgt over te gaan tot APPEND, zal hij het scherm leeg maken en komen met het schermbeeld afgebeeld in Scherm 2-3.

```
RECORD # 00008

NAAM      :                    :
ADRES     :                              :
TELEFOON  :          :
```

Scherm 2-3

Om nu een nieuw record toe te voegen, vult u dan de lege posities in, net zoals bij CREATE. HET GAAT NET ZO ALS HET INVULLEN VAN EEN FORMULIER MET EEN SCHRIJFMACHINE. U kunt net zoveel records toevoegen als u maar wilt. Wilt u APPEND beëindigen, drukt u dan op de RETURN toets wanneer de cursor op de eerste positie van het eerste veld van een nieuw record staat – net zo als bij CREATE.

Het nieuwe record is afgebeeld in Scherm 2-4. In dit voorbeeld kenden we het adres van T. E. Duim niet, zodat we op RETURN hebben gedrukt om de cursor door te laten gaan naar het veld TELEFOON.

```
RECORD # 00008

NAAM          : Duim, T. E.
ADRES         :                              :
TELEFOON      : 08373-5648:
```

Scherm 2-4

Net als in het voorbeeld bij CREATE gaat de computer automatisch door naar het volgende record (Record 9) en spoort hij u aan de gegevens daarvoor in te voeren. Drukken op de RETURN toets zal het APPENDen stoppen en de computer laten antwoorden met een puntprompt.

WIJZIGEN EN VERVANGEN

Stel nu dat we het adres van T. E. Duim krijgen en dat zijn telefoonnummer is veranderd. Er zijn twee commando's die gebruikt kunnen worden voor het wijzigen (veranderen) van velden binnen een record: EDIT en REPLACE.

Het commando EDIT werkt net zo als het commando APPEND, behalve dat het geen record toevoegt. In tegenstelling tot het commando APPEND moet u echter het RECORDNUMMER kennen. Het is een van de weinige dBASE II commando's, waarbij kennis van recordnummers is vereist. Een record kan geEDIT worden zonder die kennis, maar dan moet dat in twee stappen. Het gebruik van het EDIT commando wordt hieronder beschreven. Wilt u de EDIT functie aanroepen, typt u dan EDIT gevolgd door het gewenste recordnummer na een PUNTPROMPT.

```
.EDIT 8
```

EDIT werkt net zo als APPEND, behalve dat er geen lege VELDEN op het scherm komen, maar de inhoud van de gewenste records wordt weergegeven. Net als bij APPEND zal de cursor op de eerste positie van het eerste veld komen te staan. In dit voorbeeld gaan we het adres toevoegen en het telefoonnummer veranderen.

Om het adres te kunnen veranderen moeten we de cursor verplaatsen naar het ADRES veld. Daarvoor moeten we iets kunnen, dat we nog niet eerder hebben gedaan: de CONTROL of CTRL toets gebruiken.

DE CONTROL TOETS

De CONTROL toets wordt net zo gebruikt als de hoofdlettertoets op een schrijfmachine. De hoofdlettertoets geeft elke schrijfmachinetoets een 'tweede betekenis'. Het indrukken van de hoofdlettertoets bij het aanslaan van de 'h' zorgt op de schrijfmachine voor een hoofdletter 'H'. De CONTROL (of CTRL) toets geeft sommige toetsen een 'derde betekenis'. Wanneer bijvoorbeeld de controltoets wordt ingedrukt, zal de 'X' toets de cursor steeds één veld vooruit verplaatsen bij iedere aanslag. Dit wordt uitgesproken als CONTROL X. Vaak wordt het geschreven als X̂ (of CTRL X of CODE X). De uitwerking van enkele CONTROL toetsen is samengevat in Tabel 2-1.

Controltoets	uitwerking
X	Verplaatst de cursor 1 veld vooruit
E	Verplaatst de cursor 1 veld achteruit
D	Verplaatst de cursor 1 positie vooruit
S	Verplaatst de cursor 1 positie achteruit
Y	Wist de inhoud van het veld
V	Voegt tekens tussen op de positie van de cursor
G	Wist teken op de positie van de cursor
W	Beëindigt EDIT

Tabel 2-1 Eigenschappen van enkele controltoetsen

Nu kunnen we doorgaan met het wijzigen van RECORD 8. Aangezien we al EDIT 8 hadden getypt in antwoord op de PUNTPROMPT, staat Record 8 nu op ons scherm, met de cursor in het NAAM veld.

```
RECORD # 00008

NAAM          :Duim, T. E.          :
ADRES         :Nieuweweg 25, Renkum :
TELEFOON      :08373-135648:
```

Scherm 2-5

Om de cursor naar het veld ADRES te verplaatsen drukt u op CTRL X. De cursor gaat naar de eerste positie in het veld ADRES. Voer nu het adres in als weergegeven in Scherm 2-5. Het invoeren van return zal de cursor aan het begin van het veld TELEFOON neerzetten. Om het telefoonnummer te wijzigen typt u nu het nieuwe nummer gewoon over het oude heen.

Toen we de '8' invoerden in het telefoonnummer, verliet de computer automatisch het EDIT commando. Dit deed hij, omdat we het laatste teken hadden ingevoerd in het laatste veld van het laatste record van de database. Als dit niet het laatste veld was geweest van het laatste record, zouden we met opzet het EDIT commando hebben moeten verlaten. Dit wordt gedaan met CONTROL W. In dit geval staat de 'W' voor 'write' (schrijven). ^W verzoekt de computer de nieuwe informatie of structuur 'weg te schrijven' naar de diskette. Deze manier van stoppen met EDIT maakt duidelijk dat u klaar bent met uw veranderingen en dat ze naar diskette geschreven moeten worden voor blijvende bewaring.

Het andere wijzigingscommando, REPLACE, stelt u in staat bepaalde veranderingen aan te brengen in een of meer records, uitgaande van bepaalde criteria. Met REPLACE verandert u de inhoud van één of meerdere velden door de computer de veldnamen te geven, de nieuwe inhoud en de criteria voor het aanbrengen van de verandering.

Veronderstel bijvoorbeeld, dat de laatst ingevoerde persoon, T. E. Duim, is verhuisd. Hij heeft een nieuw adres en een nieuw telefoonnummer. Om het telefoonnummer te veranderen via het REPLACE commando, komt de transactie er als volgt uit te zien:

```
.REPLACE TELEFOON WITH'08373-2426759', ADRES WITH 'Bosweg 5,
Renkum' FOR NAAM = 'Duim, T. E.'
00001 REPLACEMENT(S)
```

Het antwoord 00001 REPLACEMENT(S) van de computer, terwijl er toch twee velden veranderd (vervangen) zijn, heeft betrekking op het aantal records dat door het commando gewijzigd is. Het commando heeft zowel het telefoonnummer als het adres door de nieuwe informatie vervangen.

Zowel EDIT als REPLACE worden op uitgebreide schaal toegepast bij het wijzigen van de inhoud van een database. EDIT is een zogenaamde schermvullende bewerking. Het databaserecord VULT zogezegd HET HELE SCHERM en kan worden veranderd door de nieuwe informatie dwars over de oude heen te typen. REPLACE kan worden gebruikt om automatisch bepaalde gegevens te vervangen door nieuwe gegevens volgens criteria die door de gebruiker zijn vastgesteld. De verandering wordt in de computer aangebracht, maar is niet voor de gebruiker zichtbaar op het scherm. Als u visuele informatie wilt hebben, moet u een afzonderlijk weergavecommando gebruiken.

De oefening met de persoonlijke telefoonlijst gaat over een vertrouwd onderwerp, we kennen het allemaal. Het is echter maar een heel beperkt voorbeeld. Het levert bij lange na niet genoeg materiaal om alle mogelijkheden van computerdatabases te demonstreren.

Daarom beginnen we met onze tweede database, waarmee we meer aspecten demonstreren van de mogelijkheden van een computerdatabase. Het gaat hier om de inventarislijst van een winkel, waarvoor we een jaarlijkse inventarisatie zullen uitvoeren, zoals ook de eigenaar van een kleine slijterij dat zou doen.

Welke zaken zouden van belang kunnen zijn voor de slijter? Dat zijn gegevens als:

Soort drank
Merknaam
Inhoud
Voorraad
Inkoopprijs
Verkoopprijs

Deze begrippen worden dan de kolommen (de velden) voor onze database. De volgende drie stappen maken deel uit van de handelingen die de vorm en de indeling van de database bepalen:

1 We wijzen VELDNAMEN toe
2 We delen ieder veld in een rubriek in (tekens, numeriek, logisch)
3 We regelen de breedte van elk veld

- In dit voorbeeld is de veldnaam voor de kolom die de soort drank bevat, DRANK. Die kolom is een tekenveld (Character), dat 10 posities breed is.
- De kolom voor de merknamen heeft MERK als veldnaam, is een tekenveld en 20 posities breed.
- Het veld voor de inhoud van de fles is INHOUD, is een tekenveld en 7 posities breed.
- De hoeveelheid die in voorraad is, heet VOORRAAD, is een getalveld (Numeric) en drie cijfers breed. Zo kunnen er bij dit winkeltje 999 flessen per flesinhoud en merk in voorraad zijn.
- De laatste twee velden, KOSTEN en PRIJS geheten, zijn ook getalvelden. Deze twee velden gaan getalwaarden bevatten, die echter op een beetje andere manier ingevoerd zullen worden. Voor beide velden nemen we 3 cijfers voor de komma en twee cijfers erachter. De maximale veldbreedte bij elk van deze twee velden is dus ZES (5 cijfers plus de decimale punt).

We hebben precies hetzelfde probleem bij het kiezen van een FILENAAM (titel) bij deze database als in het voorbeeld van de persoonlijke telefoonlijst. Inventaris Slijterij is te lang en bevat bovendien spaties. We zullen INVENTRS kiezen als filenaam. Dat is een goede omschrijving en telt acht letters.

EEN DATABASE MAKEN

Wat we net gedaan hebben, is te zien in Scherm 2-6. De database (file) INVENTRS zal in diskettedrive B: gezet worden.

```
.CREATE
ENTER FILENAME: B:INVENTRS
ENTER RECORD STRUCTURE AS FOLLOWS:
FIELD   NAME, TYPE, WIDTH, DECIMAL PLACES
001     DRANK, C, 10
002     MERK, C, 20
003     INHOUD, C, 7
004     VOORRAAD, N, 3
005     KOSTEN, N, 6, 2
006     PRIJS, N, 6, 2
007

INPUT DATA NOW? Y
```

Scherm 2-6

Na uw antwoord Y (ja) begint de computer u aan te sporen gegevens in te voeren voor het eerste record. Het schermbeeld is te zien in Scherm 2-7.

```
RECORD # 00001

DRANK      :          :
MERK       :                    :
INHOUD     :       :
VOORRAAD   :       :
KOSTEN     :       :
PRIJS      :       :
```

Scherm 2-7

We gaan nu de gegevens invoeren in de database INVENTRS. Een typisch record is afgebeeld in Scherm 2-8.

```
RECORD # 00001

DRANK      :SHERRY:
MERK       :ANDALUCIA:
INHOUD     :1.0 LTR:
VOORRAAD   : 23:
KOSTEN     : 5.59:
PRIJS      : 9.31:
```

Scherm 2-8

Dit gaat zo door, record voor record, totdat alle gegevens voor de vijftien records zijn ingevoerd. Een RETURN aan het begin van Record 16 haalt ons uit de gegevensinvoer.

OVER GEGEVENSINVOER

Het invoeren van gegevens begint bij elk veld bij de linker dubbele punt. Als het veld een tekenveld is, zullen de gegevens bij de linker dubbele punt blijven, alsof u op een stukje papier getypt had. Dit heet links aligneren.

Bij getalvelden worden de gegevens ook ingevoerd vanaf de linker dubbele punt. Wanneer u echter op de RETURN toets drukt (om de computer te vertellen, dat u klaar bent met het invoeren van het veld), gaat de computer het getal aanschuiven tegen de rechter dubbele punt, net zoals u zelf zou doen, wanneer u getallen in een kolom aan het zetten was. Dit heet rechts aligneren. De computer aligneert getalvelden rechts.

Als u probeert een letter in te voeren in een getalveld, zal dBASE II dat niet toelaten. De letter wordt niet geaccepteerd en de terminal geeft een piep-waarschuwing.

Het invoeren van gegevens kan doorgaan totdat de inventarislijst klaar is (of totdat we genoeg gegevens hebben voor ons voorbeeld). Onze voorbeelddatabase B:INVENTRS bezit 6 VELDEN en 15 RECORDS.

Om nu iets te doen met de database moeten we de computer eerst vertellen dat de database gebruikt gaat worden. In dit geval nemen we de database in gebruik door de computer de opdracht LIST B:INVENTRS te geven. De computer laat nu de hele database zien, als afgebeeld in Scherm 2-9.

```
.USE B:INVENTRS
.LIST

00001  SHERRY    ANDALUCIA      1.0 L  23  5.59   9.31
00002  SHERRY    ANDALUCIA      2.0 L   7  9.78  16.30
00003  SHERRY    ANDALUCIA      0.5 L  88  2.74   4.56
00004  WODKA     RUSSKI         1.0 L  35  3.78   6.30
00005  WODKA     RUSSKI         2.0 L   9  7.95  13.25
00006  WODKA     RUSSKI         0.5 L  75  1.49   2.48
00007  WHISKEY   SOUTHERN RYE   1.0 L  32  5.11   8.51
00008  WHISKEY   OLD IRISH      0.5 L  44  1.98   3.30
00009  WHISKEY   OLD IRISH      1.0 L  19  5.29   8.81
00010  WHISKEY   THE NEW SOUTH  1.0 L   4  7.49  12.48
00011  JENEVER   SCHIEDAM       0.5 L   5  0.99   1.65
00012  JENEVER   SCHIEDAM       0.8 L  22  1.78   2.96
00013  JENEVER   SCHIEDAM       1.0 L  21  3.50   5.83
00014  JENEVER   SCHIEDAM       2.5 L   3  6.89  11.48
00015  JENEVER   SCHIEDAM       2.0 L   5  6.47  10.78
```

Scherm 2-9

Een inventarislijst dient onder meer tot het bepalen van de hoeveelheid kapitaal, die in de voorraad is geïnvesteerd. Wanneer we een inventarislijst op de klassieke manier met potlood en papier hebben gemaakt, kunnen we de waarde van de voorraad berekenen door alle hoeveelheden te vermenigvuldigen met de overeenkomstige kosten en vervolgens de getallen op te tellen. Als de inventaris opgemaakt wordt met behulp van een microcomputer en een database management systeem als dBASE II, dan hebben we dat resultaat bijna onmiddellijk bij de hand. Het systeem zorgt daarvoor. Uit dit voorbeeld zien we voor het eerst de echte kracht van de database.

Het dBASE II commando dat de waarde van de hele voorraad vraagt, ziet er zo uit:

```
.SUM KOSTEN*VOORRAAD
1305.49
```

We hebben gevraagd om de som (het totaal) van kosten maal hoeveelheid in voorraad. Het sterretje '*' is het symbool dat de computer vraagt te vermenigvuldigen. De uitkomst van deze bewerking is meteen onder de opdracht te zien. De computer past bovenstaand commando toe op de hele database.

Stel dat u wilt weten, hoeveel kapitaal in sherry is geïnvesteerd. Die transactie is hieronder te zien.

```
.SUM KOSTEN*VOORRAAD FOR DRANK = 'SHERRY'
438.15
```

Stel nu dat u de investering wilt weten in literflessen sherry. Hieronder staan het commando en de uitkomst. In dit voorbeeld zien we nog een eigenaardigheid van het programma. AND moet worden voorafgegaan en worden gevolgd door een punt, zodat het eruit komt te zien als '.AND.'

```
.SUM KOSTEN*VOORRAAD FOR DRANK = 'SHERRY'.AND.INHOUD = '1.0 L'
128.57
```

Eenvoudige berekeningen zoals deze zijn enkel het topje van de informatie ijsberg. U kunt ze via eenvoudige databasebewerkingen tevoorschijn halen. U kunt in feite hele rapporten krijgen, zo opgezet dat u de te gebruiken informatie voor u krijgt in een vorm die aansluit bij uw behoeften. Dit alles vraagt slechts twee commando's.

STANDAARDRAPPORTEN

dBASE II beschikt over een ingebouwde functie voor het maken van rapporten, die REPORT heet. Vele rapporten kunnen alleen al uitgaande van deze voorziening gemaakt worden. REPORT kan rechtstreeks vanaf het toetsenbord van de computer voorbereid worden en het 'rapportageformulier' kan bewaard worden voor toekomstig gebruik.

Achter een PUNTPROMPT typt u REPORT. De computer zal u een reeks vragen stellen, PROMPTS genaamd. Dit is net zoiets als het invullen van een formulier. De antwoorden worden door de computer gebruikt voor het samenstellen van het rapport.

- De eerste aansporing is een verzoek om een REPORT FORM NAME. Dit is een ander soort FILENAAM, net zoiets als de FILENAAM voor de database. Filenamen voor rapportageformulieren moeten voldoen aan dezelfde regels als filenamen voor databases. Ze moeten beginnen met een diskettedrive-aanduiding, gevolgd door een dubbele punt en dan acht of minder letters en cijfers. Omdat we de database B:INVENTRS aan het gebruiken zijn, noemen we het rapportageformulier ook B:INVENTRS.

Misschien vraagt u zich af, hoe de computer het verschil kan zien tussen B:INVENTRS (de database) en B:INVENTRS (het rapportageformulier). In een database management systeem bestaan verschillende 'filetypes' en de computer heeft zijn eigen systeem om ze van labels te voorzien en zo kan hij ze blijven herkennen. Het systeem voegt .DBF (database file) toe aan de namen van files als B:TELEFOON en B:INVENTRS. Uit de stappen die we ondernemen weet het systeem in dit geval file heel goed als rapportagefile herkenbaar te maken door .FRM toe te voegen aan onze filenaam. Wij kunnen de file nog steeds B:INVENTRS noemen, omdat de computer zelf zorgt voor het maken van het onderscheid tussen B:INVENTRS.DBF en B:INVENTRS.FRM.

Nadat u de FILENAAM voor het rapport hebt ingevoerd, begint de computer u aan te sporen de informatie te verschaffen, die hij nodig heeft voor het samenstellen van het rapport. Een rapport in deze vorm is in feite een formulier dat de computer invult.

Als al eerder een rapport was gemaakt met deze filenaam op het moment dat we REPORT en vervolgens de FILENAAM opgaven, zou de computer eenvoudigweg het rapport samenstellen. We vragen nu echter voor de eerste maal een rapport en dus moeten we eerst antwoord geven op de PROMPTS van de computer om het hem mogelijk te maken het formulier samen te stellen.

- Bij de tweede prompt mogen we kiezen uit een menu met standaardkeuzemogelijkheden, namelijk de breedte van de linkerkantlijn, het aantal regels per bladzijde en de bladbreedte. Als u enkel de RETURN toets indrukt, geeft u daarmee aan, dat u niet geïnteresseerd bent in het veranderen van de gegeven instellingen. U neemt genoegen met de 'standaardindeling'.
- De prompt 'PAGE HEADING?' vraagt of u een kopregel (een titel) afgedrukt wilt hebben op elke bladzijde van het rapport.
- Als u yes antwoordt voor ja, verschijnt 'ENTER PAGE HEADING'. U zou kunnen typen INVENTARISLIJST SLIJTERIJ.

- De volgende prompt stelt u in de gelegenheid enkele (N) of dubbele (Y) regelafstand te kiezen.
- De volgende prompt vraagt of u totalen wenst.
- Zo ja (yes) dan is de volgende prompt – ARE SUBTOTALS REQUIRED?
- Zo ja (yes) dan is de volgende prompt – ENTER SUBTOTALS FIELD, dat wil zeggen, van welk veld wilt u de subtotalen? In dit geval willen we de subtotalen voor iedere soort DRANK.
- De volgende prompt is een keuze: SUMMARY REPORT ONLY? Zo nee dan wilt u een FULL REPORT. (Het voorbeeld hierna is een FULL REPORT. Een 'SUMMARY' zou alleen de soort drank geven en de subtotalen.)
- Omdat het rapport niet alleen kan worden weergegeven op het beeldscherm, maar ook kan worden afgedrukt, krijgt u de vraag, of na ieder subtotaal op een nieuwe bladzijde moet worden overgegaan. De computer vraagt u in feite, of u iedere subtotaalrubriek op een afzonderlijke bladzijde (of een afzonderlijke reeks bladzijden) afgedrukt wilt hebben. In ons voorbeeld zou bij 'yes' voor ja het rapport op vier afzonderlijke bladzijden worden afgedrukt.
- De volgende prompt vraagt om een SUBTOTAL HEADING. Dit is alleen van toepassing als een kop noodzakelijk is bij de inhoud van het veld van het subtotaal. In dit voorbeeld is geen extra tekst nodig en dus drukken we op de RETURN toets.
- Vervolgens moet de computer weten, hoeveel kolommen in het rapport moeten en wat in elke kolom moet staan. Om de benodigde informatie te krijgen, zal de computer u aansporen de breedte en de inhoud van elke kolom op te geven. De kolommen worden stuk voor stuk gedefiniëerd, gaande van links naar rechts.

De inhoud van een kolom is of een VELDNAAM of een uitdrukking als KOSTEN*VOORRAAD. Het rapport zal de inhoud van dat veld bevatten of de uitkomst van die uitdrukking.

Wanneer u klaar bent met het definiëren van alle gewenste kolommen, drukt u op de RETURN toets, wanneer de vraag gesteld wordt naar de breedte van de volgende kolom. Dit geeft het einde aan van het ontwerpen van het formulier en slaat het rapport op voor gebruik in de toekomst.

Wat we net allemaal gedaan hebben, is te zien in Scherm 2-10 hieronder.

```
.REPORT
ENTER REPORT FORM NAME: B:INVENTRS
ENTER OPTIONS, M=LEFT MARGIN, L=LINE/PAGE, W=PAGE
WIDTH
PAGE HEADING? (Y/N) Y
ENTER PAGE HEADING: INVENTARISLIJST SLIJTERIJ
DOUBLE SPACE REPORT? (Y/N) N
TOTALS REQUIRED IN REPORT? Y
SUBTOTALS IN REPORT? (Y/N) Y
ENTER SUBTOTALS FIELD: DRANK
SUMMARY REPORT ONLY? (Y/N) N
EJECT PAGE AFTER SUBTOTALS (Y/N) N
ENTER SUBTOTAL HEADING:
COL   WIDTH, CONTENTS
001   20, MERK
ENTER HEADING: MERK
002   7, INHOUD
ENTER HEADING: INHOUD
003   3, VOORRAAD
ENTER HEADING: VRD
ARE SUBTOTALS REQUIRED? (Y/N) Y
004   6, KOSTEN
ENTER HEADNG: KOSTEN
ARE TOTALS REQUIRED? (Y/N) N
005   7, KOSTEN*VOORRAAD
ENTER HEADNG: INVEST
ARE TOTALS REQUIRED? (Y/N) Y
006
```

Scherm 2-10

Nu dit voorbereidende werk klaar is, zal u de computer iedere keer dat u om een .REPORT vraagt en vervolgens de filenaam B:INVENTRS opgeeft, het rapport maken, dat is afgebeeld in Figuur 2-1. Als u hem vraagt om een .REPORT TO PRINT, in plaats van enkel om een .REPORT, zal hij zorgen voor een exemplaar op papier.

Page No. 00001 09/15/81

Inventarislijst slijterij

Merk	Inhoud	Vrd	Kosten	Invest
* SHERRY				
ANDALUCIA	1.0 L	23	5.59	128.57
ANDALUCIA	2.0 L	7	9.78	68.46
ANDALUCIA	0.5 L	88	2.74	241.12
** SUBTOTAL **		118		438.15
* WODKA				
RUSSKI	1.0 L	35	3.78	132.30
RUSSKI	2.0 L	9	7.95	71.55
RUSSKI	0.5 L	75	1.49	111.75
** SUBTOTAL **		119		315.60
* WHISKEY				
SOUTHERN RYE	1.0 L	32	5.11	163.52
OLD IRISH	0.5 L	44	1.98	87.12
OLD IRISH	1.0 L	19	5.29	100.51
THE NEW SOUTH	1.0 L	4	7.49	29.96
** SUBTOTAL **		99		381.11
* JENEVER				
SCHIEDAM	0.5 L	5	0.99	4.95
SCHIEDAM	0.8 L	22	1.78	39.16
SCHIEDAM	1.0 L	21	3.50	73.50
SCHIEDAM	2.5 L	3	6.89	20.67
SCHIEDAM	2.0 L	5	6.47	32.35
** SUBTOTAL **		56		170.63
** TOTAL **		392		1305.49

Figuur 2-1 Computerrapport van inventarislijst slijterij

Wat hebben we nou eigenlijk gedaan? Als we in het echt een kleine slijterij hadden geïnventariseerd, zouden we waarschijnlijk potlood, papier en een rekenmachine hebben gebruikt. Nu kunnen we het doen met een database management systeem op een computer. De uitkomsten zijn weinige minuten na afloop van het opmaken van de inventaris al bekend. We maken de kans op fouten een stuk kleiner door de rekenmachine niet meer te gebruiken. En dit kan allemaal zonder iets van een computer, van het programmeren ervan of van database management systemen te weten. Het enige wat we hoeven te doen, is het uitvoeren van dezelfde stappen als we in de voorafgaande bladzijden doorlopen hebben. Als je er eens goed over nadenkt, is dit een opmerkelijk voorbeeld: er komen maar twee commando's bij kijken: CREATE en REPORT.

Om nog eens op een andere manier tegen de inhoud van onze inventarislijst aan te kijken, moeten we ons overzicht even wat anders organiseren. Het kan best zijn, dat ons bewaringssysteem uitgaat van de inhoud van verschillende soorten en merken drank. Binnen de database kunnen we een index maken en sorteren. INDEXEREN en SORTEREN hebben dezelfde grondgedachte, maar worden op een verschillende manier toegepast.

INDEXEREN EN SORTEREN

Als we een database SORTEREN, geven we de databaserecords echt een andere volgorde.

Een voorbeeld van buiten de computerwereld is het sorteren van een spel kaarten. We kunnen de kaarten op een heleboel verschillende manieren sorteren, maar enkel op één manier tegelijk. Stel dat we de kaarten ordenen op hun kleuren, en vervolgens op hun waarde binnen elke kleur. Als we dit doen en we de ruitenboer willen hebben, weten we waar we moeten kijken. Als we de kaarten pakken en ze op waarde SORTEREN, hebben we een heel andere volgorde van de kaarten, maar ook nu weten we weer, waar we de ruitenboer kunnen vinden.

Een heel slimme manier om hetzelfde te bereiken, is het INDEXEREN van de kaarten. Daarbij blijft de feitelijke ordening van de database zoals die was. Bij INDEXEREN wordt een afzonderlijke lijst aangelegd die de plaats van het gewenste aangeeft. Om dit te verduidelijken kijken we opnieuw naar het spel kaarten. Op een blaadje papier maken we een lijst van alle kaarten in de volgorde die in Figuur 2-2 te zien is.

Kleur	Kaart	Plaats
Schoppen	Aas	
	Heer	
	Vrouw	
	.	
	.	
	Twee	
Harten	Aas	
	.	
	Twee	
Ruiten	.	
Klaveren	.	

Figuur 2-2

U hebt nu een geordende lijst van de 52 kaarten in het spel. Pak nu de kaarten en schud ze goed.

Pak de eerste kaart van de stapel af. Zoek de kaart op in de lijst en schrijf het getal 1 onder 'PLAATS' naast de beschrijving van de kaart (dus bijvoorbeeld Harten....Zes....1). Leg de kaart op tafel en herhaal dezelfde handeling voor alle kaarten van de stapel, zodat op de tafel een omgekeerde stapel komt te liggen.

De lijst is nu een INDEX. Iedere willekeurige kaart kan in de lijst gevonden worden en vervolgens ook in de stapel, door de kaarten op de tafel van onderaf af te tellen. Als bijvoorbeeld ruitenboer het getal 10 achter zich heeft in de lijst, is het in de stapel de tiende kaart van onderen. De nummers in de lijst, waarmee de kaarten gelocaliseerd kunnen worden, heten WIJZERS.

We kunnen een lijst maken, die een andere mogelijke ordening van de kaarten beschrijft: de kaarten gerangschikt volgens hun nominale waarde en binnen die waarde volgens hun kleur. Een dergelijke rangschikking is te zien in Figuur 2-3.

Kleur	Kaart	Plaats
Aas	Schoppen	
	Harten	
	Klaveren	
	Ruiten	
Heer	Schoppen	
	Harten	
	.	
	.	
	.	
	enzovoorts	
enzovoorts		

Figuur 2-3. Een andere lijst van de kaarten

Als we nogmaals onze stapel kaarten doorlopen en de posities van de kaarten opschrijven, hebben we uiteindelijk twee lijsten, die dezelfde feitelijke volgorde van de stapel kaarten beschrijven. Zolang we de volgorde van de kaarten niet veranderen, kunnen we kaarten snel vinden door gebruik te maken van een van beide INDEXen.

Nog een voorbeeld van INDEXen en WIJZERS is te vinden in de bibliotheek.

Als u naar de bibliotheek gaat om een heel bepaald boek te halen, kunt u het op twee manieren vinden: hetzij u gaat de rekken langs, beginnend bij de eerste kast, en gaat kast voor kast door, totdat u het boek vindt, hetzij u gebruikt de kaartenbak die u vertelt op welke schap in welke kast het boek te vinden is. De kaartenbakken vormen een meervoudige index. Er zijn INDEX kaarten op titel, op schrijver en op onderwerp. Op iedere kaart staan WIJZERS, die uitgaan van dingen die u al weet, om u te vertellen, waar het boek te vinden is.

Afhankelijk van het soort database, dat u gebruikt, kunt u een of meerdere indexen van de records aanleggen. Met het soort databases dat wij gebruiken, kunnen we uitgaande van de database B:INVENTRS bijvoorbeeld een index maken op de inhoud van de drankflessen. Wanneer we dat doen, moeten we een INDEXFILE hebben, waarin de gewenste volgorde van de records wordt opgeslagen. In dit voorbeeld noemen we de indexfile B:INHINDEX. We gebruiken eerst onze inventarisfile B:INVENTRS en kunnen dan een index maken op de inhoud van de flessen.

Deze bewerking en de uitkomst ervan zijn te zien in Scherm 2-11.

```
.USE B:INVENTRS
.INDEX ON INHOUD TO B:INHINDEX
00015 RECORDS INDEXED
.DISPLAY ALL

00003  SHERRY   ANDALUCIA      0.5 L  88  2.74  4.56
00006  WODKA    RUSSKI         0.5 L  75  1.49  2.48
00008  WHISKEY  OLD IRISH      0.5 L  44  1.98  3.30
00011  JENEVER  SCHIEDAM       0.5 L   5  0.99  1.65
00012  JENEVER  SCHIEDAM       0.8 L  22  1.78  2.96
00001  SHERRY   ANDALUCIA      1.0 L  23  5.59  9.31
00004  WODKA    RUSSKI         1.0 L  35  3.78  6.30
00007  WHISKEY  SOUTHERN RYE   1.0 L  32  5.11  8.51
00009  WHISKEY  OLD IRISH      1.0 L  19  5.29  8.81
00010  WHISKEY  THE NEW SOUTH  1.0 L   4  7.49 12.48
00013  JENEVER  SCHIEDAM       1.0 L  21  3.50  5.83
00002  SHERRY   ANDALUCIA      2.0 L   7  9.78 16.30
00005  WODKA    RUSSKI         2.0 L   9  7.95 13.25
00015  JENEVER  SCHIEDAM       2.0 L   5  6.47 10.78
00014  JENEVER  SCHIEDAM       2.5 L   3  6.89 11.48
```

Scherm 2-11

Door gebruik te maken van INDEX en een INDEX FILE laten we de eigenlijke database in de oorspronkelijke volgorde van gegevensinvoer. Dat kunt u zien door te kijken naar de kolom met recordnummers, die weergeeft waar in de database elke regel zich bevindt.

Wat hier gebeurd is, lijkt erg veel op het voorbeeld van het spel kaarten en de lijsten. De weergave wordt geregeld door de INDEX file B:INHINDEX in plaats van door de volgorde, waarin de gegevens werden ingevoerd. Een van de fijne eigenschappen van INDEX is, dat u zich niet meer hoeft te bekommeren om de volgorde, waarin records worden ingevoerd. De computer kan ze snel in bijna iedere willekeurige volgorde zetten, die u maar zou wensen, of ze op bijna elke gewenste manier groeperen.

In deze eerste twee hoofdstukken, hebben we een paar grondbegrippen en basisverrichtingen besproken van database management systemen op microcomputers. Deze basisoriëntatie en 'echte' beginervaring is de grondslag voor groter en belangrijker zaken.

Deze systemen maken gebruik van de snelheid van de computer om dingen te doen, die we in het verleden met de hand deden met behulp van 'databases op papier'. We hebben de analogie met de database op papier gebruikt om aan te geven, hoe een database management systeem werkt en om u kennis te laten maken met de terminologie.

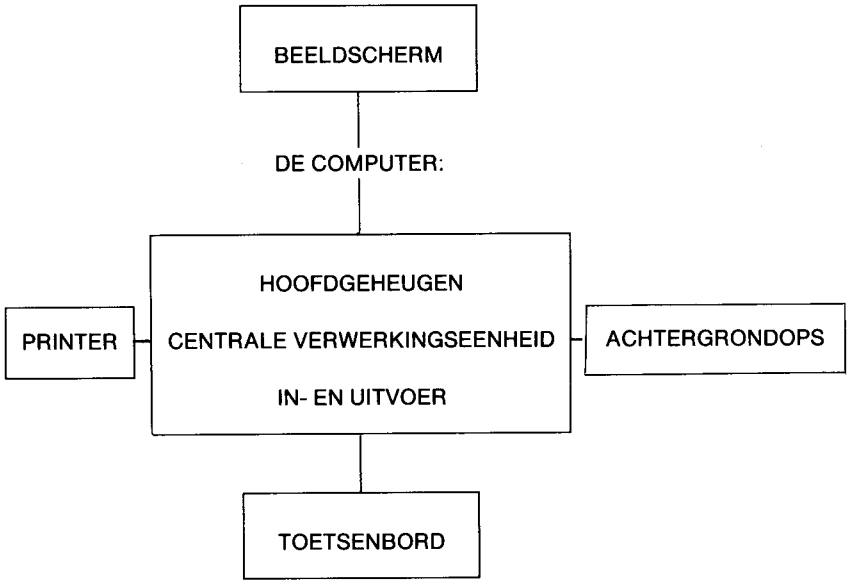
- Het database management systeem levert de middelen voor het opslaan van gegevens in de computer, voor het terughalen, bewerken en veranderen van die gegevens en voor het opstellen van rapporten die op die gegevens gebaseerd zijn.
- Bij de moderne database systemen voor microcomputers hoeft de gebruiker geen technische achtergrond te hebben of zich met technische zaken vertrouwd te maken.
- De systemen zijn er voor een groot deel in geslaagd de computer aan te passen aan de behoeften van de gebruiker in plaats van de gebruiker te dwingen zich aan te passen aan de computer. Deze systemen zullen uiteindelijk de grondslag gaan vormen voor computergebruik in het dagelijks leven.

3. HOE ZIT HET MET APPARATUUR?

De eerste twee hoofdstukken waren een verkenning van het terrein. We hebben gezien, dat een microcomputer en een database management systeem (DBMS) gemakkelijk te gebruiken zijn. Onze voorbeelden van hoe database management systemen werken en wat ze voor ons kunnen doen, zullen in de volgende hoofdstukken worden uitgebreid. We zullen dezelfde stof meer in detail behandelen en ook zullen we nieuwe onderwerpen aan u voorleggen.

In dit hoofdstuk vindt u een inleiding over microcomputerapparatuur. Om een databasesysteem te kunnen gebruiken moet u een computer hebben (hoewel u ook in staat zult zijn de uitleg te volgen en te begrijpen zonder een computer te hebben – zoals al eerder is opgemerkt). Voor het geval u niet vertrouwd bent met computers, zullen we hier de belangrijkste bestanddelen van computersystemen behandelen en vervolgens aangeven, wat u nodig hebt om een databasesysteem te gebruiken. Zoals u in dit hoofdstuk en in latere hoofdstukken zult zien, hebben uw databasebehoeften een grote invloed op de apparatuur die u het beste kunt kiezen.

In Figuur 3-1 ziet u een diagram van een typisch microcomputersysteem.



Figuur 3-1 Een typische microcomputer

Het diagram op de vorige pagina beschrijft een willekeurig computersysteem: het zou een microcomputer kunnen zijn, maar ook een heel grote computer. De computer zelf bestaat uit wat in de doos zit met het opschrift computer – de centrale verwerkingseenheid, het hoofdgeheugen en de in- en uitvoer. De andere vier dozen – beeldscherm, achtergrondopslag, printer en toetsenbord zijn randapparatuur.

DE 'COMPUTER' (bestaande uit – centrale verwerkingseenheid
hoofdgeheugen
in- en uitvoer)

De centrale verwerkingseenheid wordt meestal aangeduid als de CPU. Dit is het apparaat dat de eigenlijke bewerkingen uitvoert. Wil deze eenheid echter kunnen werken, dan moet hij de beschikking hebben over een HOOFDGEHEUGEN. Het hoofdgeheugen wordt rechtstreeks door de CPU gebruikt.

Advertenties voor microcomputers zeggen meestal zoiets als

"Koop een 24K Komkommer IV"

Die 24K hebben betrekking op de hoeveelheid intern geheugen uitgedrukt in zogenaamde BYTES. Een BYTE is de hoeveelheid geheugen die nodig is om een schrijfmachineteken als 'm' of '$' op te slaan. 24K betekent meestal 24 000. Bij een computergeheugen betekent het in feite 24 576. Een 24K computer beschikt over 24 576 BYTES aan intern geheugen.

Intern geheugen is betrekkelijk duur in verhouding tot achtergrondopslag. Wanneer u een computer koopt, zult u de hoeveelheid intern geheugen met enige zorg willen kiezen. Het databasesysteem dat u kiest zal een bepaalde minimale hoeveelheid intern geheugen nodig hebben. *U moet tenminste die hoeveelheid intern geheugen in uw computersysteem hebben zitten.*

In- en uitvoer is datgene wat de computer met de buitenwereld verbindt. In dit geval bestaat de buitenwereld uit randapparatuur. U zou redelijkerwijs kunnen aannemen, dat in- en uitvoer net zo gewoon bij de computer wordt geleverd als de banden bij een auto. Dat is niet altijd het geval – voor in- en uitvoer moet u vaak extra betalen.

Bij microcomputers is prijs echter een relatief begrip. De onderdelen van een microcomputersysteem zijn geen van alle bijzonder duur.

DE RANDAPPARATUUR

Computers hebben niet altijd randapparatuur nodig. Bij de meeste toepassingen van databases zijn echter alle randapparaten uit Figuur 3-1 nodig, op één na. Dit houdt in dat achtergrondopslag, beeldscherm en toetsenbord onontbeerlijk zijn. Bij bepaalde toepassingen is een printer ook een must. Bij andere toepassingen kan de printer eventueel ontbreken. Hij is echter altijd nuttig.

TOETSENBORDEN; BEELDSCHERMEN; TERMINALS

Het toetsenbord is een ding dat lijkt op het toetsenbord van een schrijfmachine. Het wordt gebruikt voor communicatie met de computer. De computer geeft u berichten door met behulp van het beeldscherm. Het toetsenbord en het beeldscherm kunnen los van elkaar zijn; dan komt het beeld op een televisie-achtig apparaat, dat monitor wordt genoemd. Het toetsenbord en het beeldscherm zijn ook dikwijls samengevoegd in één toestel. Dat wordt dan een computerterminal genoemd, een videoterminal, of enkel een CRT (kathodestraalbuis).

Of u een terminal hebt, dan wel een toetsenbord met monitor, hangt meestal af van het merk microcomputersysteem dat u bezit. In dit boek zullen we het voor het gemak vaak over een terminal hebben. In de praktijk is er geen verschil tussen een videoterminal en een toetsenbord met een afzonderlijke monitor. De meeste videoschermen kunnen 24 regels van 80 tekens in een keer weergeven. Sommige tonen wat minder, andere wat meer. Subjectief bezien zijn toetsenbord en beeldscherm buitengewoon belangrijk: deze apparaten zijn uw middelen om contact met de computer te hebben. U 'praat' tegen de computer via het toetsenbord; de computer 'praat' tegen u via het beeldscherm.

ACHTERGRONDOPSLAG

De achtergrondopslag is de plaats waar het grootste deel van uw informatie is opgeslagen voor gebruik door het microcomputersysteem. Zo kunt u het hoofdgeheugen van de computer goedkoop uitbreiden. Achtergrondopslag is aanmerkelijk minder duur dan hoofdgeheugen. Hij is ook aanmerkelijk langzamer (meer dan duizend maal). Wanneer de computer wordt uitgezet, wordt het hoofdgeheugen gewist, maar de achtergrondopslag niet. Achtergrondopslag kan slechts veranderd worden met een opzettelijke handeling.

Achtergrondopslag wordt gebruikt om informatie op te slaan, die u wilt bewaren, net zoals een geluidsband wordt gebruikt om geluid te 'onthouden'. Achtergrondopslagsystemen zijn zo ontworpen, dat de computer de gewenste informatie gemakkelijk kan vinden. Bij microcomputers worden twee soorten achtergrondopslag algemeen gebruikt: cassettes en diskettes.

Voor cassettesystemen zijn vaak dezelfde soort cassettebandjes nodig als bij het opnemen van geluid, of in ieder geval lijken de gebruikte cassettes op audio-cassettes. Deze systemen zijn langzaam en zijn niet geschikt voor gebruik bij databasesystemen, behalve bij bepaalde speciale toepassingen.

Diskettesystemen worden ook wel aangeduid als direct access storage devices (DASD, rechtstreeks toegankelijke opslagapparatuur). Het zijn een soort 'magnetische platenspelers'. De informatie wordt magnetisch opgeslagen op een ronddraaiende schijf in een soort groeven, net als op een plaat. Een onderdeel dat lijkt op de arm bij een platenspeler, wordt gebruikt om de diskette te 'lezen' en om erop te 'schrijven' (de arm heeft alleen geen naald, maar meer een soort magnetische kop, zoals een casetterecorder). De informatie staat op magnetische 'groeven', die in tegenstelling tot de groeven van een grammofoonplaat niet zichtbaar zijn.

Een DISKETTEDRIVE is het mechaniek dat de diskette laat ronddraaien en informatie overdraagt naar en van de diskette. De computer 'kent' de plaats van elk beetje informatie in alle diskettedrives. Deze kennis krijgt de computer door het lezen van de 'inhoudsopgave' die op iedere diskette staat. Die index op de diskette lijkt een beetje op het label van een langspeelplaat – daar staat op, welke muziek zich in iedere band op de plaat bevindt.

De diskettedrive moet bestuurd worden door een diskettecontroller, die verbonden is met de in- en uitvoer van de computer. De diskettecontroller regelt de beweging van de lees- en schrijfkop van alle diskettedrives, zodat die kop op de juiste plaats op de magneetschijf komt te staan. De diskettecontroller baant een weg voor de informatie die tussen CPU en diskette vloeit. Hij houdt de CPU ook op de hoogte van de toestand van de diskette.

Behalve diskettesystemen worden bij microcomputers ook zogenaamde vaste schijf systemen gebruikt voor dezelfde doeleinden.

DISKETTESYSTEMEN

Diskettes worden ook wel 'floppy disks' of 'flexibele schijven' genoemd. Omstreeks 1970 zijn ze door IBM ontwikkeld voor gebruik op het IBM 370 computersysteem. Dit soort schijven is het meest gebruikte soort opslagmedium voor microcomputersystemen. Een diskette kost ongeveer even veel als een cassette. De prijs van een diskettedrive ligt echter meestal veel hoger dan de prijs van een cassetterecorder.

Een diskettesysteem kan meestal veel beter werken dan een cassetterecorder. Een vaste schijf systeem doet het meestal weer een stuk beter dan een diskettesysteem. De meeste microcomputersystemen voor zakelijke en professionele toepassingen hebben een of meer diskettedrives.

Microcomputersystemen zijn doorgaans uitgerust met een tot vier diskettedrives. Die systemen die in staat zijn met database management systemen te werken, hebben er meestal minstens één. De diskette is een handige manier om nieuwe programmatuur over te brengen naar de microcomputer en om gegevens of programmatuur aan andere microcomputers te leveren.

Een diskette is een mylar film, die met oxide bedekt is en gesneden tot een schijf die lijkt op een 45 toeren plaatje. De schijf wordt verpakt in een omhulsel van plastic of papier, waarin zich gaten bevinden, dankzij welke de mylarschijf op de as van de diskettedrive kan worden geplaatst en die de lees- en schrijfkop toegang verschaffen tot de diskette. Een ander gat in de omhulling van de schijf dient voor het ontdekken van de indexmarkering. Wanneer de diskette in de diskettedrive wordt gestopt, blijft de omhulling op zijn plaats, terwijl de mylar schijf erin ronddraait. U moet ervoor zorgen, dat de mylar film schoon blijft. U mag de mylar film NIET aanraken. Aanraken van de film of blootstellen ervan aan vuil en vet kan uw gegevens ernstig verminken.

Er bestaan geen algemene standaard voor diskettesystemen. Dit kan wat problemen opleveren bij het uitwisselen van diskettes tussen verschillende systemen. Om de mogelijkheid te bieden diskettes uit te wisselen tussen verschillende systemen moeten de diskettedrives aan dezelfde standaard voldoen.

Diskettes zijn er van $3^1/_2$ inch (microfloppies), 5 inch (deze minifloppies) en van 8 inch. Van deze soorten bestaan enkelzijdige (waarbija de gegevens maar aan een kant van de schijf kunnen staan) en dubbelzijdige. De dichtheid van opgeslagen informatie is hetzij enkele dichtheid, hetzij dubbele dichtheid (tweemaal zo dicht als enkele dichtheid). Er bestaan ook systemen die viervoudige ('quad') dichtheid gebruiken.

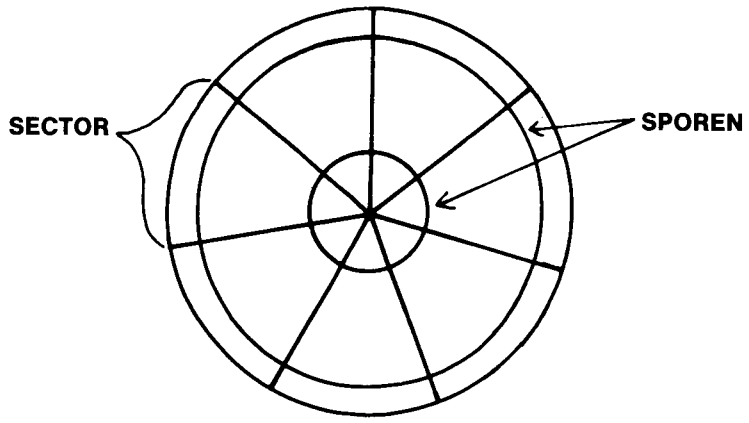
De nominale hoeveelheid informatie die op elk van deze diskettesoorten opgeslagen kan worden, staat in Tabel 3-1. Deze getallen geven een indicatie van de opslagcapaciteit van de type diskettes. Omdat de opslagdichtheid van groot belang is voor de fabrikanten van computersystemen en diskettedrives, lopen de feitelijke waarden ver uiteen. In zekere zin is dit jammer, omdat deze variatie ten koste gaat van de uitwisselbaarheid.

	5 inch	8 inch
enkelzijdig, enkele dichtheid	70 000	240 000
enkelzijdig, dubbele dichtheid	140 000	480 000
dubbelzijdig, dubbele dichtheid	280 000	960 000

Tabel 3-1 Opslagcapaciteit van diskettes in bytes

De meeste diskettes van 8 inch met enkele dichtheid hebben de indeling van de IBM 3740. Deze indeling wordt gebruikt door de diskettedrive, die de gegevens vastlegt. Oorspronkelijk werd die indeling door IBM gebruikt voor het '3740 Key to Disk Data Entry System'.

Elke '3740' diskette bevat 77 sporen (groeven) en is verdeeld in 26 sectoren. Een sector is een wig met de vorm van een taartpunt, als afgebeeld in het diagram van Figuur 3-2. Het buitenste spoor is spoor 0 en het binnenste spoor is spoor 76. Het aantal bytes, dat in een sector op een spoor opgeslagen wordt, is doorgaans onafhankelijk van de positie van het spoor. Dit houdt in, dat in de tijd waarin 100 bytes opgeslagen kunnen worden in een sector van spoor 0, ook 100 bytes opgeslagen zouden kunnen worden in een sector van spoor 76. Deze speciale indeling zorgt ervoor, dat de snelheid van gegevensoverdracht onafhankelijk is van het spoor – de buitenste sporen bewegen (uitgedrukt in inches per sekonde) sneller dan de binnenste sporen.



Figuur 3-2. Schematisch overzicht van een diskette

Voor 5 inch mini-floppy disks is de uitwisselbaarheid helaas minder goed. Voor de kleinere diskettes is er op het moment nog geen algemeen aanvaarde vastleggingsstandaard. Het uitwisselen van 5 inch diskettes tussen verschillende microcomputersystemen gaat meestal niet zo best. Het is aan te raden de uitwisselbaarheid van systemen eerst uit te proberen, voordat u grootse plannen gaat maken voor het overdragen van informatie op diskettes.

VASTE SCHIJF SYSTEMEN

Hoewel diskettes veel bij microcomputers worden toegepast, zijn zeker ook 'vaste' schijven beschikbaar en toepasbaar. 'Vaste' schijf betekent een schijfsysteem voor een microcomputer, waarbij geen diskettes worden gebruikt. En vaste schijf is niet buigzaam. Hij wordt gemaakt van een machinaal bewerkte metaalplaat. Sommige vaste schijven kunnen niet verwijderd worden. Die worden 'Winchester disks' genoemd.

Het grote verschil tussen vaste schijven en diskettes is, dat vaste schijven duidelijk betere prestaties leveren. Ze zijn ook aanzienlijk duurder. De opslagcapaciteit van vaste schijf systemen begint bij een miljoen bytes. Opslagcapaciteiten van 5 tot 40 miljoen bytes en meer zijn gebruikelijk.

Andere prestaties van schijfsystemen die u in advertenties aangeprezen kunt zien, zijn: van spoor tot spoor snelheid, gemiddelde toegangssnelheid en gegevensoverdrachtssnelheid. Dit zijn getallen die de prestaties van een bepaald apparaat aangeven.

De van-spoor-tot-spoor-tijd is de tijd die de schijfdrive nodig heeft om de kop van het ene spoor naar een aangrenzend spoor te verplaatsen. De aangeprezen waarden voor de verplaatsingstijd liggen doorgaans in de buurt van een paar milliseconden. Een milliseconde is 1/1000 seconde.

De gemiddelde toegangstijd is de tijd die gemiddeld nodig is om de kop op een nieuwe positie te zetten. Nauwkeuriger gezegd is het de tijd nodig om de kop van spoor 1 naar het laatste spoor te brengen gedeeld door 2. Voor vaste schijf systemen is die tijd meestal verscheidene milliseconden. Voor diskettesystemen wordt de toegangstijd bijna nooit in advertenties vermeld, maar hij is 100 tot 500 milliseconden. Het gaat hier om een snelheidsmaat die uiterst misleidend kan zijn en u kunt dit gegeven meestal maar beter negeren.

De overdrachtssnelheid voor gegevens is nog een eigenschap die verkopers graag gebruiken en die u zou behoren te negeren. Overdrachtssnelheden worden doorgaans opgegeven in honderdduizendtallen of miljoenen bytes per seconde.

De twee laatstgenoemde eigenschappen zijn afkomstig uit de wereld van de schijfdrives voor grote computers. Het belang ervan bij de microcomputer is twijfelachtig. Van veel groter belang voor u is, of de schijfdrive aansluitbaar is op uw computer – dat wil zeggen, wat er gedaan moet worden om het apparaat op uw computer aan te sluiten en om het dan goed te laten werken. Het aansluiten van het apparaat en het ervoor zorgen dat het goed werkt, wordt INTEGRATIE genoemd.

Gegevens op 8 inch diskettes met dubbele dichtheid worden meestal vastgelegd in IBM 34 indeling. Vastlegging in een uitwisselbare indeling is geen voldoende garantie dat de diskettes ook echt uitgewisseld kunnen worden tussen verschillende microcomputers, maar het is een begin. Een aanzienlijk percentage van de 8 inch diskettes kan tussen verschillende microcomputersystemen uitgewisseld worden.

Voor ons, in dit boek, is achtergrondopslag van groot belang. Databases worden bij microcomputers gewoonlijk opgeslagen op schijven. GEEN ENKELE wijze van database management zal werken, als u geen goede schijfopslag hebt. De huidige microcomputers kunnen alleen heel kleine databases in hun hoofdgeheugen houden.

Heel grote databases hebben meestal een vaste schijf nodig. Middelgrote databases kunnen opgeslagen zijn op diskettes. Zelfs grote databases die op vaste schijven staan, hebben vaak een 'reserve-exemplaar' op diskettes. Zo'n veiligheidskopie van de gegevens wordt een 'backup' genoemd. Een dergelijke veiligheidskopie biedt bescherming tegen ongelukken met de schijf en is een goedkoop middel om de database over te brengen van de ene microcomputer naar de andere.

DE PRINTER

Een printer is altijd nuttig, ook al hebt u er niet absoluut een nodig. Wat voor soort printer u kiest, is niet belangrijk voor de database.

Er bestaan twee verschillende soorten printers: matrixprinters en daisy-wheel printers.

- De daisy-wheel printer maakt een volledig uitgetekende letters en cijfers net als een schrijfmachine. De daisy-wheel printer wordt meestal gebruikt, wanneer de presentatie een heel goede kwaliteit moet hebben.
- De matrixprinter gebruikt een aantal kleine puntjes om het teken op te bouwen.

De afdrukkwaliteit van matrixprinters loopt nogal uiteen. Matrixprinters zijn meestal minder duur dan daisy-wheel printers en de letter is minder mooi. Doorgaans zijn ze aanmerkelijk sneller. De karakteristieke afdruksnelheid van daisy-wheel printers is 40 tot 50 tekens per seconde. De afdruksnelheid van matrixprinters is vaak meer dan 100 tekens per seconde. Afdrukken van matrixprinters komen niet altijd goed over op kopiëermachines.

HET BESTURINGSSYSTEEM

Bij de meeste computers is het zo, dat het hoofdgeheugen bij het aanzetten leeg is: de machine heeft geen doel voor ogen en kan niet veel doen. Om iets te presteren heeft een computer een BESTURINGSSYSTEEM nodig. Dit systeem is het middel waarmee u en uw database management systeem de computer gemakkelijk kunnen gebruiken en beheersen.

De meeste microcomputersystemen waarop een database management systeem gebruikt kan worden, hebben als besturingssysteem een disk operating system (DOS). Bij het aanzetten van de computer wordt dit besturingssysteem van diskette in het intern geheugen geladen.

Dit kan automatisch gebeuren, of misschien moet u er een knop op de computer voor indrukken. Het laden van het besturingssysteem heet BOOTen (opstarten). Om een database systeem te gebruiken hoeft u van het besturingssyteem niet veel te weten – nu niet en nooit niet.

In dit boek gaan we ervan uit, dat als besturingssysteem het bij microcomputers populaire CP/M wordt gebruikt. Het enige effect dat een ander besturingssysteem op dit boek zou kunnen hebben, betreft de regels voor het aanspreken van diskettedrives en/of de naamgeving van files.

EEN PAAR LAATSTE WOORDEN OVER APPARATUUR

Om goed te kunnen beginnen hebt u de juiste soort computerapparatuur nodig.

Bij een database systeem voor een microcomputer is het bestanddeel waar het het meest op aankomt, het diskette- of vaste schijf drivesysteem. Een schijfsysteem dat bij een database management systeem gebruikt wordt, moet voldoende omvang hebben om de hele database op te slaan op één enkele schijf (In Hoofdstuk 4 bespreken we, hoe u de omvang van uw database in bytes kunt vaststellen). De CPU kan alleen de gegevens gebruiken, die zich in de aangesloten schijfdrives bevinden. Die gegevens zijn, zoals dat heet, ON-LINE beschikbaar. Als u diskettedrives hebt, zijn alleen de diskettes die werkelijk in de diskettedrives gestopt zijn ON-LINE. Als u een computer bezit en die wilt gebruiken voor een databasetoepassing, zult u goed op de hoogte moeten zijn van uw schijfmogelijkheden.

Als u nog geen computer gekozen hebt, dan is het voor u en uw portemonnee het beste als volgt te handelen: kies het database systeem, bepaal de eisen die u aan de opslag moet stellen en kies dan een computersysteem dat (1) past bij de programmatuur van uw database systeem en (2) uw opslageisen aankan en over een aanvaardbare mogelijkheid beschikt om meer achtergrondopslag toe te voegen, mocht die nodig worden.

Deel 2

DEEL TWEE

In Deel Twee gaan we ons verder verdiepen in het plannen en gebruiken van databases. We beginnen met het bespreken van de eenvoud en het belang van een goed ontwerp, daarna maken, wijzigen en onderhouden we databases die echt wat om het lijf hebben en tenslotte gaan we ze gebruiken. We bouwen voort op de grondslagen en de voorbeelden die vertrouwd zijn uit Deel Een.

4. UW DATABASE VOORBEREIDEN

Planning wordt vaak gezien als iets vervelends, in het bijzonder door de beginneling. Niet alleen bij de databasebeginner is dat zo. Helaas geldt echter, dat als u niet eerst een voldoende goed ontwerp maakt, u wel eens ongelukkig zou kunnen zijn met het resultaat en in het ergste geval alles zou moeten overdoen.

Neem nog eens het voorbeeld van het maken van een database op papier. Als degene die typt, de layout van de kolommen niet goed ontwerpt, zal de 'database' waarschijnlijk over de rechterkantlijn van het papier heengaan. Dit is geen ramp, maar het is vervelend en het werk zal overgedaan moeten worden.

Hetzelfde geldt voor onze computerdatabase. Hij zal niet over de rand van het papier heengaan, maar als hij niet goed ontworpen is, moet u misschien op uw schreden terugkeren en extra werk doen om de dingen te krijgen, zoals u ze wilt hebben. Eén heel goede manier om een ontwerp te maken voor een database is hem op te zetten met in uw achterhoofd de gedachte, dat u het werk nog een- of tweemaal zult moeten overdoen. Deze werkwijze heet ITERATIEVE VERBETERING. Een van de voordelen van het werken met een computerdatabase (afgezet tegen het werken op papier) is, dat u werkelijk ingrijpende veranderingen kunt aanbrengen in de database zonder de gegevens opnieuw te hoeven invoeren. De computer kan, in de meeste gevallen althans, gebruikt worden om de meerderheid van de gegevens – zo niet alle – terug te halen, die opgeslagen waren voordat de verandering werd aangebracht.

De eerste stap bij het ontwerpen van uw database is te weten wat u ermee wilt gaan doen. Dan kunt u besluiten, wat u erin gaat zetten. De volgende stap is een lijst maken van de dingen die erin moeten. Probeert u niet het in één keer volmaakt te doen. Bijna alle tekortkomingen kunnen gemakkelijk hersteld of overwonnen worden.

Laten we als eerste voorbeeld weer eens kijken naar de slijterijdatabase die we in Hoofdstuk 2 gemaakt hebben. Deze 'inventarislijstdatabase' heeft als bedoeling ons te vertellen, hoeveel er in voorraad is en wat de voorraad waard is. Bij het maken van deze database somden we gewoon de opschriften van de kolommen op om de opbouw van de records vast te leggen. Als we een ontwerp wilden maken voor deze database, zouden we een lijst maken van de dingen die erin moesten komen:

Merk drank
Inhoud van de fles
Soort drank
Verkoopprijs
Inkoopprijs
Voorraad ervan

Deze lijst geeft weer, wat opgenomen werd in de inventarislijst voor de slijterij in Hoofdstuk 2. Sinds we zijn begonnen die op te zetten, hebben we tenminste één extra ding bedacht, dat eigenlijk ook in de lijst opgenomen zou moeten worden:

Plaats in de winkel

Het ontbreken hiervan is gemakkelijk weer goed te maken. We zullen stap voor stap aangeven, wat er gedaan moet worden, en u zult zien, hoe gemakkelijk het veranderen in zijn werk gaat. Dat zou u echt gerust moeten stellen.

U hebt al een database B:INVENTRS en daarin zitten een heleboel gegevens. Wat gaat u nu doen? U moet een veld toevoegen en u wilt geen gegevens verliezen, die u al hebt ingevoerd.

GEEN PROBLEEM! U kunt dit vanaf uw toetsenbord in een paar minuten oplossen zonder de reeds ingebrachte gegevens te verliezen. Er is trouwens meer dan één oplossing – meer dan één manier om dit te doen.

OPLOSSING #1: MAAK EEN NIEUWE FILE

Eén mogelijke oplossing is het maken van een nieuwe file (die noemen we B:NIEWFILE) die dezelfde opbouw heeft (veldnamen, veldsoorten en –breedten) als B:INVENTRS maar waarin een nieuw veld is toegevoegd voor de plaats in de winkel. We maken B:NIEWFILE door gebruik te maken van het dBASE II commando CREATE. Zodra de nieuwe file gemaakt is, gaan we alle informatie die nu is opgeslagen in B:INVENTRS, in de nieuwe file zetten.

Hoe het maken in zijn werk gaat, is te zien in Scherm 4-1. Merkt u op, dat we ervoor gekozen hebben het nieuwe veld PLAATS ergens midden in het record te zetten. Dit hebben we gedaan om duidelijk te maken, dat het database management systeem werkt met velden – de plaats van het veld binnen het record is niet van belang.

```
.CREATE
ENTER FILENAME: B:NIEWFILE
ENTER RECORD STRUCTURE AS FOLLOWS:

FIELD          NAME, TYPE, WIDTH, DECIMAL PLACES
001            DRANK, C, 10
002            MERK, C, 20
003            INHOUD, C, 7
004            VOORRAAD, N, 3
005            PLAATS, C, 10
006            KOSTEN, N, 6, 2
007            PRIJS, N, 6, 2
008

INPUT DATA NOW? N
```

Scherm 4-1

Ons besluit is ditmaal NEE, we willen nu geen gegevens invoeren. Het invoeren van 'nee' haalt u uit de database die u net gemaakt hebt. Om weer met deze database te gaan werken, moet u het commando USE toepassen. Alle gegevens opgeslagen in de file B:INVENTRS kunnen worden toegevoegd aan de file B:NIEWFILE via de dialoog die is afgebeeld in Scherm 4-2.

```
.USE B:NIEWFILE
.APPEND FROM B:INVENTRS
00015 RECORDS ADDED
```

Scherm 4-2

In het korte voorbeeld hierboven werd een nieuwe databasefile gemaakt via CREATE. Alle informatie opgeslagen in de databasefile B:INVENTRS wordt toegevoegd aan de nieuwe file (B:NIEWFILE) door gebruik van het dBASE II commando APPEND. Alle informatie komt in de juiste velden te staan, ook al werd het nieuwe veld – PLAATS – 'middenin' de database gezet.

We hebben nu twee bestanden , B:NIEWFILE en B:INVENTRS. Op het moment bevatten ze beide precies dezelfde informatie. B:NIEWFILE bevat het nieuwe veld – PLAATS. We zouden willen dat deze file B:INVENTRS heette. Aan de oude B:INVENTRS hebben we nu niets meer. De oude B:INVENTRS kunnen we wegdoen en B:NIEWFILE kunnen we de naam B:INVENTRS geven met de dialoog afgebeeld in Scherm 4-3.

```
.DELETE FILE B:INVENTRS
FILE HAS BEEN DELETED
.RENAME B:NIEWFILE TO B:INVENTRS
```

Scherm 4-3

OPLOSSING #2: PAS DE STRUCTUUR AAN

dBASE II stelt u in de gelegenheid de database te veranderen op een manier die lijkt op EDIT. Dit wordt bereikt met een commando MODIFY STRUCTURE. Het is echter helaas zo, dat wanneer we de structuur op deze manier 'wijzigen' alle gegevens vernietigd worden. Daarom voeren we het 'wijzigen' uit door eerst een kopie te maken van de structuur en vervolgens deze kopie van de structuur te wijzigen. De oorspronkelijke database is dan nog ongeschonden en onveranderd. We geven nu de commando's weergegeven in Scherm 4-4 hieronder.

```
.USE B:INVENTRS
.COPY STRUCTURE TO B:NIEWFILE
.USE B:NIEWFILE
.MODIFY STRUCTURE
MODIFY ERASES ALL DATA RECORDS...PROCEED(Y/N) Y
```

Scherm 4-4

U kunt nu veilig de structuur van B:NIEWFILE aanpassen, zonder dat dat invloed heeft op de oorspronkelijke database B:INVENTRS. Scherm 4-5 is een duplicaat van onze oorspronkelijke database B:INVENTRS. Hoewel deze database er hetzelfde uitziet als B:INVENTRS, is hij in feite de kopie ervan, B:NIEWFILE. B:INVENTRS bestaat nog steeds – op veilige afstand van onze aanstaande structuurverandering.

NAME	TYPE	LEN	DEC	
FIELD 01:DRANK	C	010	000	:
FIELD 02:MERK	C	020	000	:
FIELD 03:INHOUD	C	007	000	:
FIELD 04:VOORRAAD	N	003	000	:
FIELD 05:KOSTEN	N	006	002	:
FIELD 06:PRIJS	N	006	002	:
FIELD 07:				:
FIELD 08:				:
FIELD 09:				:
FIELD 10:				:
FIELD 11:				:
FIELD 12:				:
FIELD 13:				:
FIELD 14:				:
FIELD 15:				:
FIELD 16:				:

Scherm 4-5

In dit voorbeeld gaan we een nieuw veld tussenvoegen tussen VOORRAAD en KOSTEN. Om dit voor elkaar te krijgen drukken we vier maal op de RETURN toets. Daarmee komt de cursor aan het begin van de veldnaam KOSTEN te staan. Vervolgens houdt u de CONTROL toets ingedrukt en slaat u de 'N' aan. Nu wordt een blanco regel tussengevoegd op veld 5. Vervolgens typt u de nieuwe veldnaam in, het veldtype en de veldbreedte.

Het schermbeeld ziet er nu uit als Scherm 4-6 hieronder.

NAME	TYPE	LEN	DEC	
FIELD 01:DRANK	C	010	000	:
FIELD 02:MERK	C	020	000	:
FIELD 03:INHOUD	C	007	000	:
FIELD 04:VOORRAAD	N	003	000	:
FIELD 05:PLAATS	C	010	000	:
FIELD 06:KOSTEN	N	006	002	:
FIELD 07:PRIJS	N	006	002	:
FIELD 08:				:
FIELD 09:				:
FIELD 10:				:
FIELD 11:				:
FIELD 12:				:
FIELD 13:				:
FIELD 14:				:
FIELD 15:				:
FIELD 16:				:

Scherm 4-6

Druk nu op CONTROL W. De database B:NIEWFILE beschikt nu over het nieuwe veld dat u wilde tussenvoegen. De al in B:INVENTRS aanwezige gegevens kunnen worden toegevoegd en de file kan een andere naam krijgen, net zoals in het vorige voorbeeld (Scherm 4-2 en Scherm 4-3). Voert u dit uit door de gegevens van B:INVENTRS toe te voegen:

```
.APPEND FROM B:INVENTRS
00015 RECORDS APPENDED
```

Op dit moment hebt u zowel de oude als de nieuwe versie van uw database. Het is het beste de nieuwe database eerst even te bekijken om zeker te weten dat alles goed zit, voordat u de oude versie vernietigt.

```
.LIST
```

Wanneer het antwoord van de computer op 'list' op het scherm verschijnt, zult u een extra ruimte zien tussen de kolommen voorraad en kosten.

Deze open ruimte moet plaats gaan bieden aan de inhoud van het nieuwe veld. Als alles er goed uitziet, kunt u de oude file B:INVENTRS wegdoen. Gewoon:

```
.DELETE FILE B:INVENTRS
FILE HAS BEEN DELETED
.RENAME B:NIEWFILE TO B:INVENTRS
```

Daarmee is oplossing # =2 klaar: de structuur van uw database is aangepast.

U ziet, het is vrij eenvoudig; eigenlijk is er niets aan. U kunt in het begin (bij CREATE) in de database zetten wat u goeddunkt, en daarna kunt u via ITERATIEVE VERBETERING uw benadering verfijnen naarmate u een beter inzicht krijgt in wat u doet, en u zich bewust wordt van dingen die u niet voorzien had, toen u begon.

Laten we nu even een stukje teruggaan – terug naar de planning voordat u uw database begint te maken en te gebruiken. U bent nu zover dat wij u wat over de werking en de beperkingen van databases kunnen vertellen en dat is ook nodig.

Als u op papier met een schrijfmachine een database zou gaan opzetten, zou u twee dingen doen – iedere kolom een opschrift geven en uitzoeken hoeveel posities u elke kolom in beslag zou laten nemen. Dezelfde twee dingen moet u doen voor een computerdatabase. Bovendien maakt het bepalen van de soort informatie per elke kolom deel uit van de planning.

In computerdatabases worden drie soorten velden gebruikt. Dat zijn:

TEKENVELDEN (CHARACTER)
GETALVELDEN (NUMERIC)
LOGISCHE VELDEN

TEKENVELDEN zijn de meest voorkomende soort. Een tekenveld kan alles bevatten dat op een schrijfmachine getypt kan worden. Dat zijn letters (zowel kleine letters als hoofdletters), cijfers en bijzondere tekens als ?, &, < en verder de spatie. Een tekenveld kan doorgaans voor alles en nog wat gebruikt worden. We kunnen eigenlijk elk veld in de database wel tot tekenveld maken. Een paar kenmerkende tekenvelden zijn NAAM, ADRES en TELEFOON uit het voorbeeld van de telefoonlijst in Hoofdstuk 1.

De afmeting (breedte) van een tekenveld is het aantal schrijfmachineposities dat nodig zou zijn voor het langste gegeven voor dat veld. Elke letter, elk cijfer, elk speciaal symbool en elke 'spatie' telt als één teken. Elk teken neemt één BYTE aan geheugen in beslag. Telkens wanneer we de veldbreedte in verband brachten met ruimte op een getypte bladzijde, hebben we die vergeleken met de ruimte in het geheugen van de computer. Veldbreedte wordt altijd weergegeven in BYTES. Het aantal bytes is hetzelfde als het aantal posities dat nodig is om het veld op een getypte bladzijde te zetten.

GETALVELDEN kunnen alleen getallen bevatten. Ze worden meestal alleen toegepast, wanneer de getallen erin gebruikt gaan worden voor berekeningen. Ze kunnen hetzij gehele getallen bevatten, hetzij niet-gehele getallen. Behalve cijfers als 1 en 8 kunnen ze decimale punten bevatten en mintekens. Een negatief getal als – 281.65 neemt 7 posities in beslag (7 BYTES) en heeft twee decimale posities. Bij de meeste database systemen wordt het plusteken stilzwijgend aangenomen en hoeft het niet te worden ingevoerd. Het minteken en de decimale punt nemen elk een positie in en moeten worden meegeteld, wanneer de veldbreedte bepaald wordt.

Getalvelden worden verder 'rechts gealigneerd' door de computer. Daarom worden ze ook vaak gebruikt voor getallen die niet in berekeningen worden gebruikt, maar die wel rechts gealigneerd moeten worden. Tekenvelden worden door de computer gewoonlijk 'links gealigneerd'. Voorbeelden van links en rechts gealigneerde velden zijn te zien in Figuur 4-1.

Links gealigneerd	Rechts gealigneerd
1	1
10	10
100	100

Figuur 4-1 Alignering

LOGISCHE VELDEN worden gebruikt wanneer er slechts twee mogelijkheden zijn voor de gegevens. Rekeningen zijn hetzij betaald, hetzij niet betaald. Studenten hebben een cursus wel of niet gevolgd. U leest dit of u leest dit niet. Logische velden nemen altijd één BYTE (één positie) in beslag. De gegevens kunt u invoeren als T of F (voor TRUE=waar of FALSE=onwaar), of ook als Y of N (voor Yes=ja of No=nee).

Vlak na het bepalen welk 'type' veld u wenst, komt het toewijzen van een veldnaam aan de beurt. Uit Hoofdstuk 1 weten we dat de veldnaam uit hoogstens 10 tekens mag bestaan. Veldnamen moeten zo worden gekozen, dat ze een goede beschrijving geven, maar voor het gemak ook zo kort mogelijk zijn. Stel bijvoorbeeld eens, dat u velden zou gaan maken, die de omvang van een voorraad weergeven voor iedere maand van het jaar. Als u van typen houdt, kunt u de velden namen kunnen geven als JANUARI, FEBRUARI enzovoorts. Maar net zo goed zijn JAN, FEB enzovoorts.

De volgende beslissing die u moet nemen, heeft alleen betekenis voor getalvelden: krijgt het veld niet-gehele getallen en zo ja, hoeveel decimale posities hebt u maximaal nodig achter de punt. Die beslissing is niet bepaald moeilijk en we zullen zien, hoe die wordt genomen, wanneer we het ontwerpen nogmaals doorlopen – waarbij we ditmaal de structuur op ons beeldscherm maken.

We gaan het ontwerp – een 'databaseontwerp' – uitwerken voor de voorraaddatabase die we in Hoofdstuk 2 al gemaakt hebben. Die heet nu B:INVENTRS en is voorzien van een nieuw veld voor de plaats van de artikelen. We gaan dezelfde veldnamen, veldsoorten en – breedten toepassen, als we in het voorbeeld gebruikt hebben. Het nieuwe veld met de plaats van een artikel zou PLAATS genoemd kunnen worden. We maken er een tekenveld van. Voor dit voorbeeld geven we het veld de willekeurige breedte van 10 posities.

Een ontwerp voor de database B:INVENTRS zou er ongeveer uitzien als Figuur 4-2. Merkt u op, dat die figuur zelf op een database lijkt. Als u meerdere databases hebt, is het vaak slim een database te hebben die de ontwerpen voor de databases bevat. Een dergelijke ontwerpendatabase wordt wel een DATA DICTIONARY genoemd.

Beschrijving veld	Naam	Type	Breedte	Decimalen
Soort drank	DRANK	C	10	
Merk drank	MERK	C	20	
Inhoud van de fles	INHOUD	C	7	
Verkoopprijs	PRIJS	N	6	2
Inkoopprijs	KOSTEN	N	6	2
Voorraad ervan	VOORRAAD	N	3	
Plaats in de winkel	PLAATS	C	10	

TOTAAL AANTAL BYTES 62
VERWACHT AANTAL RECORDS 1000

Figuur 4-2 Voorbeelddatabaseontwerp voor B:INVENTRS

Deze database heeft zeven velden en neemt 62 bytes aan geheugen voor elk record. Voordat u redeneert: 'ach, wat zou het, waarom zou ik dit allemaal gaan uitschrijven, ik kan dat wel uit mijn hoofd' – moet u bedenken, dat uw database wel eens een paar dozijn velden zou kunnen bevatten met honderden bytes per record. In zo'n geval moet u uw ontwerp vergelijken met de middelen die u ter beschikking staan. Alle database systemen hebben beperkingen, ook uw computer heeft die en daarmee moet u rekening houden als uw toepassing werkelijk groot wordt.

Wat de computer betreft, ligt de beperking bij databasesystemen voor microcomputers voornamelijk in de capaciteit van de diskettedrive(s). Deze beperking kan doorgaans worden verschoven door hetzij meer diskettedrives toe te voegen, hetzij schijfdrives met een grotere capaciteit aan te sluiten.

Het is bemoedigend op te merken, dat de omvang van de database vaak aansluit bij de beschikbare financiële middelen. Degenen die grote databasebehoeften hebben, hebben dikwijls dienovereenkomstige middelen. Beperkingen in de achtergrondopslag kunnen vaak worden opgelost met apparatuur in de prijsklasse van duizend tot ruim tienduizend gulden.

De beperkingen die het database systeem zelf oplegt, zijn interessanter. Deze beperkingen van en grenzen aan database systemen zouden u gemakkelijk kunnen verleiden tot een zoektocht naar een of ander nieuw, prachtig database management systeem dat al uw problemen zou oplossen.

De grenzen van de mogelijkheden van een database systeem betreffen typisch de volgende zaken:

Aantal velden
Veldbreedte
Aantal bytes in een record
Aantal records in een file (database)

Bij dBASE II is het maximum 32 velden, waarbij ieder veld beperkt moet blijven tot 254 bytes en elk record tot 1000 bytes. Het aantal records is beperkt tot 65 535. Dit houdt in, dat de maximale omvang van een dBASE II database file 65 535 000 bytes is. Dat is zo groot, dat het onwaarschijnlijk is, dat u deze grenzen ooit zult bereiken.

Om u een indruk te geven van wat wij met groot bedoelen, kunt u ervan uitgaan dat zo'n database bij gebruikmaking van een gewone standaard-schrijfmachine, A4 papier, een kantlijn van 2 centimeter en pica-letters meer dan 18 670 bladzijden zou beslaan. Een microcomputer met een Winchester Disk kan een database 'lezen' met een snelheid van 16 000 tekens per seconde. Dat betekent, dat het de computer meer dan een uur zou kosten om zo'n database alleen al te 'lezen'. Wanneer u databases met deze omvang behandelt, zult u waarschijnlijk nog andere beperkingen ondervinden, hetzij in de computerapparatuur, hetzij in het besturingssysteem. Het veelgebruikte besturingssysteem voor microcomputers CP/M 2.2 kent bijvoorbeeld een maximale filegrootte van 8 000 000 bytes.

Eén van de meest voorkomende problemen bij het ontwerpen van een database is, dat een ontwerp meer velden heeft, dan het database systeem verschaft. Daar is een gemakkelijke oplossing voor. Splits het ontwerp gewoon op in twee of meer databases. Als u dat doet, wordt elke database een 'file' binnen een grotere database en misschien zou u elk van deze databases een file moeten noemen. Alle files samen zijn dan eigenlijk pas de database.

Wanneer u dit zo doet, moet u een manier vinden om de databases te koppelen. Eén manier om dit te doen, is te zorgen voor een of meer gemeenschappelijke velden in elke database file. Laten we bijvoorbeeld eens kijken naar een database voor een basisschool. Ons denkbeeldige databaseontwerp zou 60 velden kunnen hebben en eruit kunnen zien als Figuur 4-3.

Veld	Omschrijving	Veldnaam	Type	Breedte	Decimalen
1	naam leerling	NAAM	C	30	
2	klas	KLAS	C	3	
3	lokaal	LOKAAL	C	1	
4	leerkracht	LEERKRACHT	C	15	
5	zittengebleven	ZITTENBL	L	1	
–	–	–	–	–	
–	–	–	–	–	
57	adres	ADRES	C	30	
58	telefoonnummer	TELEFOON	C	8	
59	in noodgeval	NOODNAAM	C	30	
60	telefoonnummer	NOODTEL	C	8	
TOTAAL AANTAL BYTES				341	
VERWACHT AANTAL RECORDS				600	

Figuur 4-3 Databaseontwerp voor basisschool

Dit is een heel goed voorbeeld van een soort database die meestal behoorlijk groot is. Hij heeft een achtergrondopslag van iets meer dan 200 000 bytes (tekens) nodig, terwijl men zegt dat meer dan 90% van alle databases minder dan 100 000 tekens beslaan. Hier komt iets naar voren, dat zelfs door grote instellingen met gespecialiseerd personeel vaak over het hoofd wordt gezien:

U MOET UW TOEPASSING BESTUDEREN, VOORDAT U APPARATUUR EN PROGRAMMATUUR KOOPT.

Bij deze toepassing zijn wat betreft de apparatuur twee diskettedrives vereist, waarbij tenminste één van de drives in staat is ongeveer 500 000 bytes op te slaan. Misschien kunt u met minder toe, maar dan wordt u misschien beperkt in dingen die u later zou willen doen. Zou u bijvoorbeeld een veld aan de database willen toevoegen, dan hebt u 400 000 bytes nodig om de twee databases op te slaan op het moment dat u alle records uit de ene database naar de andere hebt overgebracht (zie bladzijde 68 en volgende).

De enige beperking aan de databaseprogrammatuur (bij dBASE II) is in dit voorbeeld het aantal velden: er zijn meer velden dan één databasefile aankan. De oplossing is de databasefile op te splitsen in twee of meer files. Wanneer databases worden gesplitst en 2 of meer files gaan innemen, moeten we de files koppelen. Dit wordt gedaan door gebruikmaking van het 'gemeenschappelijke element' – een bepaald gegeven, dat in de hele database in dezelfde vorm voorkomt, bijvoorbeeld de naam van de leerling.

Dit is niets nieuws. Het is heel waarschijnlijk dat de inhoud van het mapje van elke leerling uit meer dan één blad papier bestaat. In een dergelijk mapje zou de naam van de leerling doorgaans ook op elk blad papier staan. In onze database krijgt ieder van de twee files het NAAM veld, dat de naam van de leerling bevat. Het totale aantal velden is nu 61 en het totale aantal tekens is 371. Er zijn 61 velden, omdat we NAAM opnieuw hebben toegevoegd in veld 33 om aan te geven, bij wie de informatie hoort in de velden 34-60. Daarmee hebben we het noodzakelijke gemeenschappelijke element ingevoegd na het afsplitsen van de velden 34-60.

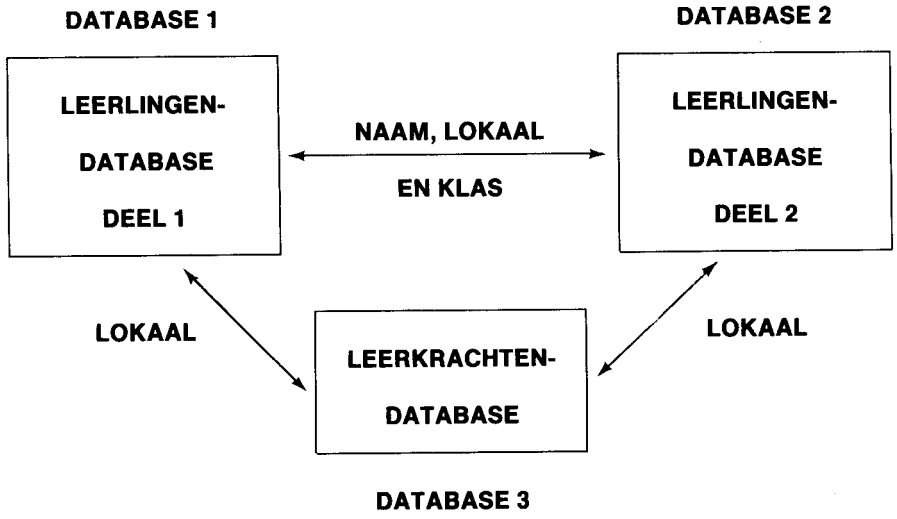
Er ontstaat alleen een probleem, wanneer er twee leerlingen op school zitten met dezelfde naam. Dat is helemaal niet zo onwaarschijnlijk. Wij pakken dit probleem aan door beide files drie velden gemeenschappelijk te laten hebben – NAAM, KLAS en LOKAAL. Het ontwerp heeft nu 63 velden nodig met 375 bytes per 'record'. Bij gebruik van dBASE II kan ons ontwerp worden uitgevoerd met 2 bestanden die samen de volledige database vormen.

Nog een mogelijke oplossing is het toewijzen van een herkenningsnummer aan elke leerling. Dit heeft bepaalde voordelen. Er zijn minder bytes nodig dan de aanbevolen oplossing en er zijn maar twee extra velden vereist in plaats van drie. Wel moet dan iemand de extra moeite nemen om iedere leerling een eigen herkenningsnummer te geven.

Laten we nu eens kijken naar de velden KLAS, LOKAAL en LEERKRACHT. Op vele scholen zou dit (drie)dubbele informatie zijn. Meneer Jansen zou normaal alleen lesgeven in lokaal 102 aan klas 6. Als dit het geval is, is het slim en doelmatig een derde databasefile in te stellen met informatie over de leerkrachten. Omdat de school waarschijnlijk toch al informatie bijhoudt over het personeel, kan zo een veld weggelaten worden uit de leerlingenfile. In dit geval bespaart dat ongeveer 9000 bytes aan geheugen (600 records x 15 bytes) – plus vooral de moeite om 600 namen te typen en alles te veranderen, wanneer een leerkracht vervangen wordt.

De schooldatabase bestaat nu uit drie files. De files staan in een bepaald verband tot elkaar. Dit is tussen twee haakjes de echte definitie van een RELATIONEEL database systeem. Files kunnen verband met elkaar houden en zo wordt nodeloze duplicatie voorkomen. De technische term voor een databasefile is eigenlijk RELATIE. Het is doorgaans verstandig informatie die bijeen gebruikt wordt, in één relatie te groeperen. Het is immers eenvoudiger met één database te werken, in plaats van met twee of meer.

In ons voorbeeld zijn de drie databasefiles (de drie relaties) met elkaar verbonden, zoals afgebeeld in Figuur 4-4. Deze Figuur toont de manier waarop files met elkaar verbonden kunnen worden.



Figuur 4-4

Een ander aspect waarmee bij het ontwerpen rekening gehouden moet worden, wordt getoond in Figuur 4-5.

NAAM	ACHTERNAAM
	VOORNAAM
	TUSSENVOEGSEL
ADRES	STRAAT
	NUMMER
	POSTCODE
	PLAATS
	PROVINCIE
	LAND
TELEFOON	NETNUMMER
	ABONNEE

Figuur 4-5. Twee mogelijke verzamelingen velden

Dit Figuur wil duidelijk maken dat één veld veel kan inhouden. In het eerste geval hebben we drie velden, in het tweede elf. Uw toepassing houdt waarschijnlijk het midden daartussen. Om uit te maken, of gegevens samengevoegd moeten worden in een enkel veld of niet, moet u inzicht hebben in het toekomstige gebruik van de gegevens. Een vuistregel is, dat als gegevens zelden of nooit afzonderlijk worden gebruikt, ze gecombineerd moeten worden. Het samenvoegen van gegevens als achternaam, voornaam en voorletters in een enkel veld naam, maakt vaak een efficiënter gebruik van ruimte mogelijk. En het is in ieder geval gemakkelijker uit te voeren.

De lijst in Figuur 4-5 betreft precies dezelfde informatie die we behandeld hebben in het oorspronkelijke voorbeeld van de telefoonlijst. Er is hier echter informatie bij, die we in het oorspronkelijke voorbeeld hebben weggelaten, bijvoorbeeld de postcode. De postcode zou ondergebracht kunnen worden in het veld adres van het oorspronkelijke voorbeeld. Of u een veld toevoegt voor de postcode of die opneemt in het adres, hangt ervan af hoe u van plan bent de informatie te gaan gebruiken.

Naarmate uw database groter wordt, kost het de computer steeds meer tijd hem te doorzoeken. Als de database maar klein is, hoeft u helemaal niet voorzichtig te zijn met de ruimte die wordt toegewezen aan een veld. Maar bij een grote database wilt u wellicht de afmeting van elk veld nauwkeurig bezien – en zelfs bekijken of u al dan niet een veld voor iets uittrekt.

Denkt u erom: hoewel we het aantal schrijfmachineposities gebruikt hebben om toelichting te geven op het aantal bytes, dat een veld innam, zal een getypte bladzijde toch veel lege ruimte bevatten, die gebruikt wordt om kolommen te scheiden.

DOET U DIT NIET IN EEN COMPUTERDATABASE

Ongebruikte bytes zijn niet nodig om velden te 'scheiden'. Als het veld de leeftijd van een leerling bevat en de hoogst mogelijke leeftijd 9 is, gebruikt u dan enkel één byte voor dat veld. Het scheiden van velden wanneer ze getoond worden, hetzij op de terminal, hetzij op een printer, is een afzonderlijke kwestie die in een later hoofdstuk besproken wordt.

Zoals we in het begin al opmerkten, wordt het maken van een ontwerp vaak als een vervelend karwei gezien. De drang om te gaan beginnen is soms allesoverheersend. Naarmate u meer ervaring krijgt, zult u ook begrip krijgen voor de onschatbare waarde van een goeddoordachte planning. Planning dwingt u over het probleem na te denken voordat u iets doet. Als het iets vervelends lijkt, bedenkt u dan dat niet voorbereiden ertoe kan leiden dat u werk moet overdoen, wat nog veel vervelender is. Het is beter over planning te denken als iteratieve verbetering (dat wil zeggen verbetering door herhaalde pogingen).

1 Begin met een bruikbaar 'raamontwerp'
2 Bouw daarop voort, totdat u het systeem zover af hebt, dat u het op de computer kunt zetten

De gevaarlijkste valkuil waar u in kunt lopen, is het zoeken naar volmaaktheid. Dit kan ongelofelijk duur worden en kan u afhouden van succes bij het voltooien van een bruikbaar databaseontwerp. Pas dus op voor perfectionisme!

Misschien bent u niet vertrouwd met database management systemen voor computers, maar de grondgedachte, het opzetten en het gebruik van computerdatabases is beslist niet moeilijk. Er zijn een heleboel vergelijkbare voorbeelden uit de wereld van potlood en papier, waarmee u wel vertrouwd bent. Doet u het rustig aan en legt u eens de nodige verbindingen met uw schat aan ervaringen. Een computerdatabase is een gemakkelijke stap naar uw toekomst. Het is bepaald geen onbeduidende stap en bovendien nog een gemakkelijke.

5. UW DATABASE OPZETTEN

Wanneer u de plannen af hebt, bent u zover dat u uw database kunt beginnen 'op te zetten'. Dit begint met het opzetten van het raamwerk voor de database, dat we uitvoerig besproken hebben in Hoofdstuk 4. Het wordt afgerond door alle gegevens in te voeren in dit raamwerk – record voor record. Het enige probleem, dat u bij dit invoeren van gegevens zult tegenkomen, is waarschijnlijk stierlijke verveling.

Voordat microcomputers en hun database management systemen ontwikkeld werden, was het maken van een file niet eenvoudig. Het duurde lang en kostte veel, er was dure apparatuur nodig en beroepsprogrammeurs moesten opdraven. U had natuurlijk zelf kunnen leren programmeren, maar tot voor kort kon u niet om de aanschaf van dure apparatuur heen. Nu vandaag de dag database management systemen en goedkope microcomputerapparatuur beschikbaar zijn, kunt u zonder moeite alles zelf doen. En tenzij u erg langzaam typt, gaat het nog snel ook.

DEEL EEN IN HET KORT

Het invoeren van gegevens werd in Deel Een verduidelijkt door twee voorbeelddatabases op te zetten, B:TELEFOON en B:INVENTRS. In deze voorbeelden hebben we gezien dat het in elkaar zetten van het raamwerk voor een database in twee stappen gebeurt.

> EERST kiest u een filenaam (een titel) voor de database.
>
> DAN definieert u stuk voor stuk alle VELDEN (alle kolommen) in de database.

De regels voor het kiezen van een filenaam worden vastgelegd door het besturingssysteem van de computer. Verschillende microcomputers kunnen verschillende besturingssystemen gebruiken. De regels voor het kiezen van filenamen kunnen dus van computer tot computer verschillen. Omdat wij bij de voorbeelden in dit boek gebruik maken van CP/M – een algemeen gebruikt besturingssysteem – wordt alles wat met filenamen van doen heeft, besproken in termen van de regels die bij CP/M gelden. Als u een computer hebt, die geen CP/M gebruikt, of een dergelijke computer op het oog hebt, kunt u het handboek van die computer raadplegen voor de regels voor filenamen.

Bij CP/M mag een filenaam bestaan uit hooguit acht letters en cijfers. Hij moet beginnen met een letter. Hij mag geen spaties bevatten. Hier volgen enige voorbeelden van goede filenamen:

```
HFDSTK1
SCHOOL
TELEFOON
```

Voorbeelden van VERKEERDE filenamen zijn:

HOOFDSTUK 1	Te lang en bevat een spatie
BLABLABLABLA	Te lang
1HFDSTK	Begint met een cijfer

Het doel van de filenaam is de computer duidelijk te maken, met welke file u wilt werken. Als de computer meer dan één diskettedrive heeft, moet u doorgaans het kenmerk van een bepaalde diskettedrive toevoegen voor aan de filenaam. Bij CP/M worden diskettedrives en vaste schijf drives aangeduid met een letter gevolgd door een dubbele punt (bijvoorbeeld A:). Volgens het voorafgaande zijn de volgende filenamen goed:

A:HFDSTK1	File bevindt zich in drive 'A'
C:SCHOOL	File bevindt zich in drive 'C'
B:TELEFOON	File bevindt zich in drive 'B'

De diskettedrive-aanduiding is geen echt bestanddeel van de filenaam. Die aanduiding moet namelijk veranderen, als de diskette in een andere diskettedrive wordt gedaan. Als we bijvoorbeeld de diskette met de file HFDSTK1 uit drive A halen en in drive B stoppen, zal de file aangeduid moeten worden als B:HFDSTK1.

U mag meer dan één databasefile hebben met dezelfde naam, zolang die files zich maar niet in dezelfde drive bevinden. Het systeem maakt het trouwens niet mogelijk twee databasefiles met dezelfde naam op één en dezelfde diskette te hebben. Een poging daartoe wordt dan ook bestraft met het geheel wissen van het bestand dat als eerste op de diskette stond.

Een database is een bepaald soort FILE, een '.DBF' file. Er bestaan andere filesoorten, waarmee de computer kan werken. In Deel Een hebben we even met een ander soort file gewerkt, een zogenaamde FORMULIERfile ('.FRM' file), die we gebruikt hebben om het voorbeeldrapport over de inventarisatie van de slijterij te maken. Een databasefile en een formulierfile mogen op dezelfde diskette staan en dezelfde filenaam hebben. Wanneer een databasefile gemaakt wordt, hangt dBASE II automatisch '.DBF' achter de filenaam. Wanneer een rapport wordt gemaakt, wordt automatisch '.FRM' achter de filenaam toegevoegd. '.DBF' en '.FRM' zijn voorbeelden van filetypes. De filenaam wordt bepaald door u, de gebruiker. Bij dBASE II wordt het filetype bepaald door het systeem. Het filetype is belangrijk voor het systeem en het gebruikt die informatie bij het uitvoeren van zijn taken.

Zodra u een filenaam hebt gekozen voor uw database, bent u zover dat u de structuur van de database kunt gaan vastleggen. De informatie die de computer nodig heeft, is:

1 Het aantal VELDEN (kolommen).

2 De naam van elke kolom.

3 De breedte van elke kolom (dat wil zeggen het aantal tekens of cijfers).

4 De soort informatie in elk van de kolommen (cijfers (N), tekens (C) of een logische waarde (L))

Het database management systeem zal u door middel van een zogenaamde aansporing op het juiste moment om ieder gegeven vragen, dat het nodig heeft.

```
.CREATE
ENTER FILENAME: B:INVENTRS
ENTER RECORD STRUCTURE AS FOLLOWS:
FIELD       NAME, TYPE, WIDTH, DECIMAL PLACES

001         DRANK, C, 10
002         MERK, C, 20
003         INHOUD, C, 7
004         VOORRAAD, N, 3
005         KOSTEN, N, 6, 2
006         PRIJS, N, 6, 2
007

INPUT DATA NOW? (Y/N) N
```

Scherm 5-1

We hebben nu een database gemaakt, die zich in diskettedrive B bevindt, met INVENTRS als filenaam. Wat we tot nu toe gedaan hebben, komt ruwweg overeen met een nieuwe map pakken, 'INVENTARIS' op het etiket schrijven, de opschriften van kolommen op blanco papier typen, lijnen op het papier trekken om de kolommen van elkaar te scheiden, het papier in de map doen en vervolgens de map in de archiefkast leggen.

HETZELFDE EN MEER

Bij het invoeren van de informatie afgebeeld in scherm 5-1 zou u wel eens een typefout kunnen maken. Laten we aannemen, dat u de tekst INHOUD voor veld 3 verkeerd hebt gespeld. Als we de fout opmerken, voordat we zijn doorgegaan naar veld 4, kunnen we hem verbeteren. Op de meeste toetsenborden zit een toets met het opschrift RUB of DEL. Het indrukken van deze toets zal het teken links van de cursor wissen en de cursor één positie naar links brengen. De typefout kan verbeterd worden door alles te wissen tot aan het punt van de fout en vervolgens de invoer vanaf dat punt opnieuw te typen.

Maar veronderstel nu, dat u een fout ontdekt, als u al naar een ander veld gegaan bent – u kunt nu niet even terug om de fout te herstellen. Gaat u dus door met het definiëren van de overige velden. Druk dan op de N toets, wanneer de computer vraagt, of u gegevens wilt invoeren.

Type achter de puntaansporing MODIFY STRUCTURE. Een schermbeeld als Scherm 5-2 zal op uw terminal verschijnen. De fout kan worden hersteld door de cursor tot het onjuiste veld te laten afdalen, de juiste informatie in te typen en vervolgens CONTROL W aan te slaan (W voor 'write' (schrijven) vertelt de computer: 'Klaar met wijzigen, schrijf wat ik gedaan heb, weg naar diskette.')

```
NAME              TYPE  LEN  DEC
FIELD 01:DRANK    C     010  000  :
FIELD 02:MERK     C     020  000  :
FIELD 03:INHOUD   C     007  000  :
FIELD 04:VOORRAAD N     003  000  :
FIELD 05:KOSTEN   N     006  002  :
FIELD 06:PRIJS    N     006  002  :
FIELD 07:                          :
FIELD 08:                          :
FIELD 09:                          :
FIELD 10:                          :
FIELD 11:                          :
FIELD 12:                          :
FIELD 13:                          :
FIELD 14:                          :
FIELD 15:                          :
FIELD 16:                          :
```

Scherm 5-2

Om gegevens te kunnen invoeren in uw nieuwe database, moet u eerst de databasefile openen. In dBASE II krijgt u dit voor elkaar met het commando USE. In ons voorbeeld zou dit als volgt gaan:

USE B: INVENTRS

Het invoeren van gegevens begint met het commando APPEND.

.APPEND

```
RECORD #00001

DRANK      :                    :
MERK       :                         :
INHOUD     :          :
VOORRAAD   :    :
KOSTEN     :       :
PRIJS      :       :
```

Scherm 5-3

De computer zal u nu gaan aansporen voor het eerste record, als afgebeeld in Scherm 5-3. Het commando APPEND voegt in feite records toe aan het einde van de database. Als de database al 49 records had, zou het antwoord op een APPEND commando gelijk zijn aan Scherm 5-3, behalve dat in dat geval het recordnummer 00050 zou zijn.

Wanneer u alle gegevens voor een record hebt ingevoerd, als afgebeeld in Scherm 5-4, zal de computer automatisch het scherm schoonmaken en beginnen u aan te sporen gegevens in te voeren voor het volgende record.

```
RECORD #00001

DRANK      :SHERRY     :
MERK       :ANDALUCIA          :
INHOUD     :1.0 LTR    :
VOORRAAD   : 23:
KOSTEN     : 5.59:
PRIJS      : 9.31:
```

Scherm 5-4

OVER GEGEVENSINVOER

Het nut van de database hangt voor een groot deel af van de kwaliteit ervan. U hebt er weinig aan de belastinginspecteur te kunnen vertellen 'Ik weet, dat er fouten inzitten, maar het is wel heel vlug klaar!' Omdat het invoeren van gegevens vaak vervelend en eentonig werk is, is het verleidelijk en zelfs algemeen gebruikelijk, dit deel van het werk te laten verrichten door de goedkoopste werkkracht, die maar te vinden is. Dit is GEEN goed idee.

In veel gevallen zal een reusachtige hoeveelheid gegevens in de nieuwe database ingevoerd moeten worden. De meeste slijterijen hebben bijvoorbeeld meer dan onze 15 artikelen in voorraad. Het invoeren van de gegevens zal verreweg het meest tijdrovende deel van het databasegebruik zijn. Het invoeren gebeurt met de snelheid van een mens; het terughalen met de snelheid van een computer.

Bovendien is dit het deel van het werk, waarbij de meeste fouten gemaakt zullen worden. Het invoeren van een enkel record, of zelfs van verscheidene records, kan zonder fouten gebeuren. Maar het is onwaarschijnlijk, dat het invoeren honderden en misschien wel duizenden records lang zonder fouten kan doorgaan.

FOUTEN BIJ GEGEVENSINVOER

Wanneer u een groot aantal nieuwe records aan een database toevoegt, zoals in het begin zal gebeuren, is het mogelijk, dat u door een of andere oorzaak het spoor bijster raakt. Misschien bedenkt u, terwijl u druk bezig bent met het invoeren van gegevens, dat er misschien iets verkeerd is ingevoerd in het vorige record. U kunt niet gewoon 'even teruggaan' om te veranderen, wat fout is. U moet dan APPEND verlaten. Dit krijgt u voor elkaar door helemaal aan het begin van het eerste veld op de returntoets te drukken. Is het record, waarin u aan het werk bent, al grotendeels juist, maakt u dat dan eerst af en verlaat u append bij de prompt die het begin van het volgende record aangeeft. Als het record waarmee u bezig bent werkelijk een puinhoop is, dan kunt u ontsnappen door te drukken op CONTROL Q, wat u niet alleen uit de APPEND werkstand haalt, maar ook het foute record verwijdert, waaraan u bezig was.

Als u eenmaal gestopt bent met APPEND, kunt u het vorige record gaan bekijken door gebruik te maken van DISPLAY of EDIT. Beide zullen ze u het laatste record tonen, dat u had ingevoerd. EDIT geeft de gegevens in precies dezelfde vorm weer als APPEND dat deed. EDIT stelt u ook in staat allerlei fouten in het record te verbeteren.

Om fouten echt te kunnen verbeteren, moeten we wat weten over de verplaatsing van de cursor. U moet de computer op de plaats van de fout kunnen neerzetten, voordat u die fout kunt gaan verbeteren.

Fouten kunnen gemakkelijk verbeterd worden door de cursor terug te brengen naar de plaats van de fout en de goede informatie dan in te typen. Op veel terminals zitten vier toetsen met pijltjessymbolen erop. Met deze toetsen kunt u de cursor verplaatsen bij het invoeren van gegevens in een database. De pijltjes 'omhoog' en 'omlaag' horen de cursor een veld terug of vooruit te zetten in een record. De pijltjes 'naar rechts' en 'naar links' zouden de cursor één tekenpositie naar rechts en naar links moeten kunnen verplaatsen.

Sommige terminals beschikken echter niet over zulke toetsen voor de besturing van de cursor of misschien werken die toetsen niet bij het databasesysteem. In dat geval kunt u de controltoets + een van de toetsen E, S, D of X gebruiken om de cursor te verplaatsen. U herinnert zich onze bespreking van de CONTROL (CTRL) toets nog wel uit Deel Een: hij stelt u in staat de meeste lettertoetsen een 'derde betekenis' te geven.

Bij dBASE II wordt deze derde betekenis gebruikt om de cursor te besturen en om andere hulp te geven bij het wijzigen van gegevens – zoals u hierna kunt lezen.

De cursor wordt verplaatst met behulp van de CONTROL toets. Het symbool ^duidt de CONTROL toets aan. ^D betekent: houdt de CONTROL toets ingedrukt, terwijl u de D toets aanslaat. Dit zal er bij dBASE II voor zorgen, dat de cursor één positie naar rechts gaat. Evenzo zal ^ S de cursor één positie naar links verplaatsen. ^E breng de cursor één veld terug (dus in de richting van het begin van het record). ^X brengt de cursor één veld vooruit. De CONTROL toets kan samen met de toetsen E, S, D en X gebruikt worden om de cursor te verplaatsen, zodat u fouten kunt verbeteren of gewoon het ingevoerde kunt veranderen, wanneer dat nodig is.

Dankzij het verplaatsen van de cursor kunt u nu de fouten verbeteren, die u gemaakt hebt. Onthoud hierbij, dat EDIT u terug brengt naar eerder ingevoerde records zodat u verbeteringen kunt aanbrengen. Control W zal die verbeteringen 'wegschrijven' naar de diskette.

Het verbeteren van een fout kan zelfs nog gemakkelijker gedaan worden, als u de fout op tijd opmerkt. De computer is buitengewoon verdraagzaam jegens slecht typewerk, in de war raken, wat u maar uithaalt – tot het moment waarop de gegevens voorgoed worden opgeslagen. Dat opslaan gebeurt pas, wanneer u een record verlaat en een volgend binnengaat. Het verlaten vriest de informatie in. Tot dan toe is niets in het record blijvend opgeslagen. U kunt zich dus naar believen verplaatsen en veranderen, ZOLANG U ZICH NOG IN HET RECORD BEVINDT. Het is daarom aan te raden, wat u hebt ingevoerd, snel nog even door te kijken, voordat u doorgaat met het volgende record.

Laten we bij wijze van voorbeeld wat correcties aanbrengen. Stel dat u KASSTT hebt ingevoerd, terwijl u KAST wilde hebben. De laatste T kan worden verwijderd door de cursor op de laatste T te zetten en op de spatiebalk te drukken. De S die teveel is, kan op twee manieren worden verwijderd: zet de cursor op de tweede S en druk dan of op de RUB (of DEL) toets, of gebruik CONTROL G (^G). RUB veegt het teken uit links van de cursor (en ook de ruimte die het innam). CONTROL G wist het teken waarop de cursor staat. In beide gevallen is het eindresultaat KAST.

U had natuurlijk ook gewoon over het woord KASSTT heen kunnen typen en de spatiebalk kunnen gebruiken om de twee extra tekens op het einde te laten verdwijnen. Gewone toetsenbordtekens kunnen altijd daar op het scherm gezet worden, waar de cursor staat. Het doet er niet toe, of daar al tekens stonden of niet. Het laatst ingevoerde teken 'wint'.

Tegenover het verwijderen van tekens staat het toevoegen van tekens. Stel dat u KAT had getypt, toen u KAST wilde hebben. U wilt de letter S tussenvoegen tussen de A en de T. Dit kan worden bereikt door de cursor op de laatste T te zetten. U voegt dan de S in door aan te slaan ^V, dan S en tenslotte opnieuw ^V.

^V is het 'invoegcommando'. ^V is een zogenaamd 'wisselcommando'. U bent ongetwijfeld vertrouwd met wisseltoetsen, die of 'AAN' zijn of 'UIT'. Een wisselcommando werkt net zo, in zoverre dat het een toestand weergeeft, die of 'AAN' of 'UIT' is. Dezelfde ^V zorgt voor zowel het aan- als het uitzetten. Als het commando AAN is en u typt ^V, dan zal het UIT-gaan, en omgekeerd. Zolang het commando aan is, kunt u zoveel tekens tussenvoegen als u wilt. Het zal AAN blijven, totdat u ^V aanslaat en het zo UIT zet.

Bij gegevensinvoer komt nog iets te pas – naast het verbeteren van fouten – waarvoor u wellicht belangstelling hebt. Misschien hebt u maar een deel van de informatie beschikbaar, die nodig is om een record helemaal te vullen, maar wilt u toch hetgeen u wel hebt al invoeren.

Als u maar een deel van de informatie voor een record wilt invoeren, doet u dat dan gerust en drukt u vervolgens op ^C. Dit zal u rechtstreeks naar het volgende record brengen, zonder dat u alle overige velden hoeft te doorlopen.

In de laatste paar alinea's hebben we een aantal nuttige CONTROL-toetsfuncties genoemd (^G=teken wissen, ^V=invoegen, ^Q=doe de gegevens van dit record weg, waarvan ik een puinhoop gemaakt heb, zodat ik overnieuw kan beginnen).

De CONTROL toets levert nog een heleboel andere hulpmiddelen. De huidige voorbeelden hebben de bedoeling u vertrouwd te maken met het algemene principe en met enkele van de belangrijkste toepassingen.

Tot nu toe is het opzetten van een database helemaal rechttoe rechtaan gegaan. U zet een raamwerk op voor een file volgens eenvoudige regels en vervolgens voert u de gegevens in. Het invoeren van de gegevens gaat voort, totdat u alle gegevens hebt ingevoerd. Dan is de database klaar om voor iets nuttigs gebruikt te worden – bijvoorbeeld om u de informatie te leveren, die nodig is bij het leiden van een bedrijf.

Als er maar weinig gegevens ingevoerd hoeven te worden, dan is de eenvoudige rechttoe rechtaan manier die besproken is, waarschijnlijk de 'beste' manier. U gaat op eenvoudige wijze recht op het doel af.

Als er heel veel gegevens ingevoerd moeten worden, dan kan het handig zijn manieren te vinden, waarop het computersysteem echt kan helpen bij het invoeren.

INVOERHULP VAN DE COMPUTER

Er bestaan een paar ingebouwde hulpmiddelen bij het invoeren van gegevens en een paar eenvoudige procedures, die u zelf kunt schrijven, die de computer in staat stellen te helpen bij de gegevensinvoer. In de rest van dit hoofdstuk, zullen we drie dergelijke hulpmiddelen bespreken:

1 EIGEN AANSPORINGEN

2 SET CARRY ON/OFF

3 MENUSYSTEMEN

EIGEN AANSPORINGEN en MENUSYSTEMEN kunt u maken via eenvoudige procedures, die u zelf zult leren schrijven overeenkomstig uw behoeften. Een 'procedure' is een gemakkelijke manier om de computer speciale dingen voor u te laten doen. Wat er allemaal komt kijken bij het 'leren' van een procedure aan de computer, zal worden besproken in Deel Vier, dat begint met Hoofdstuk 9. Maar het gebruik van de eindproducten van een procedure – eigen aansporingen en menusystemen – wordt hier besproken, omdat ze het invoeren van gegevens zoveel gemakkelijker maken. 'SET CARRY ON/OFF' is een ingebouwd hulpmiddel, dat u met een eenvoudig commando kunt aanzetten.

EIGEN AANSPORINGEN

EIGEN AANSPORINGEN zijn een voorbeeld van een bepaald soort hulp, dat de computer kan verlenen. In ons voorbeeld van de slijterij beschrijven de veldnamen aardig de inhoud van de velden. Dit is vaak – maar niet altijd – het geval. Wanneer dit niet het geval is, dan zou het aardig zijn meer informatie te kunnen opnemen. We zouden bijvoorbeeld de aansporing er ongeveer zo uit kunnen laten zien:

```
VOER HET SOORT DRANK IN (SHERRY, WHISKEY, ENZ):
```

Dit zegt veel meer dan enkel het laten zien van het woord DRANK. In het algemeen komt dit soort hulp van de computer zeer van pas. Het is erg nuttig als u een heleboel gegevens hebt en u iemand wilt laten helpen bij het invoeren ervan. Dit geldt in het bijzonder als de veldnamen niet zo erg veel zeggen over wat in de velden moet staan.

Zoals we hiervoor al gezegd hebben, kunnen eigen aansporingen als deze betrekkelijk eenvoudig verzorgd worden door middel van PROCEDURES.

Procedures hebben, net als databases, filenamen. Bij dBASE II worden procedures COMMANDO files genoemd. Voor de namen van COMMANDO files gelden dezelfde regels als voor andere filenamen. U moet ook aangeven, in welke diskettedrive de procedure zich bevindt.

Om snel te laten zien, hoe een eigen prompt zou kunnen werken, zullen we even aannemen dat er al een procedure geschreven is, die zorgt voor duidelijke prompts. We zullen deze voorbeeldprocedure B:INVOER noemen. Om de computer de procedure te laten uitvoeren in dBASE II voert u in

```
.DO B:INVOER
```

achter een puntprompt. DO B:INVOER werkt net zo als APPEND. APPEND zou een leeg 'standaardrecord' tevoorschijn halen om in te vullen; B:INVOER haalt een leeg 'zelfontworpen recordformulier' tevoorschijn. DO B:INVOER maakt Scherm 5-5.

```
RECORD #00001

VOER DE SOORT DRANK IN (WODKA, COLA, ENZ) :                    :
VOER DE MERKNAAM IN                       :                          :
VOER DE FLESINHOUD IN (BIJV. 1.0 L)       :          :
AANTAL FLESSEN DAT HIERVAN IN VOORRAAD IS:      :
KOSTEN VAN INKOOP   :          :
 PRIJS IN VERKOOP :          :
```

Scherm 5-5

We merken tussendoor even op, dat het schrijven van procedures voor database management systemen helemaal niet moeilijk is. Het kan zelfs nog leuk zijn ook.

We laten hetgeen u de computer kunt leren doen, nu even rusten en gaan het hebben over iets dat de computer zelf al kan.

SET CARRY ON/OFF

Bij veel databases moeten een heleboel gegevens meerdere malen worden getypt. Op een school zijn er veel meer leerlingen dan lokalen en leerkrachten. In het voorbeeld van de slijterij is ieder soort drank van meerdere merken verkrijgbaar. In veel gevallen kunnen de gegevens op zo'n manier gegroepeerd worden dat de hoeveelheid gegevens die ingetypt moet worden, verminderd wordt. In de slijterij is de voorraad in de rekken gegroepeerd op de soort drank – Wodka, Jenever, enz. – voor het gemak van de klant. De verschillende flesinhouden van eenzelfde merk zijn meestal bij dat merk gegroepeerd.

Wanneer de meerdere malen in te voeren gegevens bij elkaar staan, zoals in deze twee voorbeelden, kan de computer de hoeveelheid typewerk verminderen door die gegevens steeds 'mee te nemen' uit het vorige record. Bij dBASE II wordt dit meenemen 'aangezet' door het commando SET CARRY ON. Het wordt weer afgezet door SET CARRY OFF.

Laten we nu eens kijken, hoe het invoeren van gegevens in zijn werk zou gaan als CARRY aangezet was. Met APPEND krijgen we eerst het beginscherm en voeren we de gegevens in voor Record 00001.

```
RECORD #00001

DRANK       :SHERRY     :
MERK        :ANDALUCIA          :
INHOUD      :1.0 LTR:
VOORRAAD    :23:
KOSTEN      :5.59:
PRIJS       :9.31:
```

Scherm 5-4

Normaal zou vervolgens het promptscherm verschijnen, helemaal blanco, met RECORD #00002 erboven. Met CARRY ON, zal het schermbeeld er precies zo uitzien als Scherm 5-4, met alleen het recordnummer verhoogd tot 00002. Alles wat we hoeven te doen, is die velden te veranderen, die verschillen van Record 1. Wanneer we verder gaan, zal record 3 er net zo uitzien als record 2 op het moment dat we daarnaartoe doorgingen. Deze hulpmogelijkheid kan twee dingen bereiken: de hoeveelheid typewerk kan erdoor verminderen en daarmee ook de hoeveelheid typefouten.

EEN MENUSYSTEEM

Nu terug naar iets anders dat u de computer kunt 'leren' doen. Bij deze werkwijze wordt een procedure geschreven, die degene die de gegevens invoert, een stel 'keuzemogelijkheden' aanbiedt. Dit heet een MENUSYSTEEM. Het brengt de kans op typefouten tot een minimum terug.

Helaas wordt de kans op ernstiger fouten groter. U kunt deze werkwijze goed toepassen, wanneer de gegevens in opeenvolgende records weinig gemeen hebben en CARRY ON dus weinig waarde heeft.

Stel eens dat we een procedure geschreven hebben om via een MENU de inventarislijst voor onze slijterij te maken. In ons denkbeeldige voorbeeld zou op de terminal het schermbeeld van Scherm 5-6 te zien kunnen zijn.

```
                KIES UIT DE VOLGENDE MOGELIJKHEDEN

A – 0.25 LITER                  1 – SHERRY
B – 0.50 LITER                  2 – JENEVER
C – 0.80 LITER                  3 – COLA
D – 0.90 LITER                  4 – WHISKEY
E – 1.00 LITER                  5 – WODKA
F – 1.80 LITER                  6 – CHAMPAGNE
G – 2.00 LITER                  7 – RUM
H – 2.50 LITER                  8 – BERENBURG
I – MANDFLES                    9 – OVERIGE

VOER KEUZE FLESINHOUD IN UIT LINKER KOLOM  :  :

VOER SOORT DRANK IN UIT RECHTER KOLOM  :  :

VOER MERKNAAM IN  :                  :

VOER VOORRAAD IN  :  :

VOER VERKOOPPRIJS IN  :      :

VOER INKOOPKOSTEN IN  :  :
```

Scherm 5-6

Het bovenstaande voorbeeld maakt niet alleen gebruik van de nuttige kanten van onze bespreking van eigen aansporingen hiervoor, maar ook van de mogelijkheid van keuze uit een menu. De informatie '2.5 liter cola' wordt ingevoerd als H en 3.

In dit voorbeeld neemt het menu en het invoeren maar één scherm in beslag. Als meer gegevens moeten worden ingevoerd, hebben we soms meer dan één schermbeeld nodig. Als er bijvoorbeeld zowel voor de soort drank als voor de flesinhoud 30 tot 40 keuzemogelijkheden waren, zouden we voor elk van de twee keuzekolommen alleen al een eigen scherm willen gebruiken. Dit zou dan onmiddellijk tot drie opeenvolgende schermbeelden hebben geleid.

Menuprocedures en andere uitgewerkte promptsystemen komen ook goed van pas wanneer u een heleboel gegevens door anderen wilt laten invoeren en u niet van plan bent de aard van de gegevens of het bedienen van het databasesysteem iedere keer weer uit te leggen. Er bestaan gevallen waarin alles wat ingevoerd moet worden, met menu's te kiezen is.

VOORWAARDELIJK VERVANGEN

Behalve deze drie nogal voor de hand liggende hulpmiddelen bij het invoeren van gegevens, bestaat er nog een wat subtieler methode voor het invoeren van bepaalde soorten informatie 'onder bepaalde voorwaarden'.

Het voorbeeld dat hoort bij deze methode van VOORWAARDELIJK VERVANGEN, is afkomstig uit de database met leerlingen van een basisschool.

De database bevat de NAAM, het LOKAAL, de KLAS en de LEERKRACHT voor elke leerling. Bij het 'opbouwen' van de database hoeven alleen de NAAM, het LOKAAL en de KLAS te worden ingevoerd. Als dat voor elk kind gebeurd is, kan de informatie over de LEERKRACHT toegevoegd worden met een bewerking als het dBASE II commando REPLACE.

```
.REPLACE LEERKRACHT WITH 'HR. JANSEN' FOR LOKAAL = '102'
.REPLACE LEERKRACHT WITH 'MW. DE VRIES' FOR LOKAAL = '103'
ENZ
```

Het fijne aan het opzetten van een database is, dat het maar één keer hoeft te gebeuren, meestal tenminste. Die opbouw – het invoeren van de gegevens – wordt soms het LADEN van de database genoemd. Het opzetten is bepalend voor al het toekomstige gebruik van de database. Als de informatie niet volledig is, of als de informatie niet foutloos wordt ingevoerd, zal het uiteindelijke resultaat onvolledig of onjuist zijn. En u zult dat niet leuk vinden. Pas als de gegevens wel correct zijn ingevoerd, zult u er veel aan hebben – en daar is het allemaal om begonnen.

De mogelijkheden die u ter beschikking staan bij het 'opbouwen' van de uitgangsdatabase variëren van het eenvoudige en voor de hand liggende APPEND tot steeds beter ontwikkelde procedures voor verduidelijkende prompts. De volgende hulpmiddelen staan u ter beschikking om uw inspanningen zo klein mogelijk te laten zijn:

- gegevens overbrengen naar ieder volgend record (CARRY),
- keuze uit MENU's en
- VOORWAARDELIJK VERVANGEN.

Deze hulpmiddelen verminderen de hoeveelheid typewerk die nodig is. Hoe groter en ingewikkelder uw database, des te meer zullen procedures voor u van belang zijn.

6. UW DATABASE WIJZIGEN EN ONDERHOUDEN

Als een database eenmaal is opgebouwd, is het onvermijdelijk, dat hij veranderd zal worden. Allerlei veranderingen maken het wijzigen van records noodzakelijk. Behalve dat records van binnen veranderen, bijvoorbeeld in de velden VOORRAAD en PRIJS, moeten bepaalde records toegevoegd en andere gewist worden. Als voorschriften van de overheid veranderen, moeten misschien nieuwe velden toegevoegd worden aan de database. Deze belangrijke dagelijkse bezigheid – het veranderen van de database – heet BIJWERKEN.

Het bijwerken van de database is vast en zeker datgene dat de meeste tijd kost. Dat is voornamelijk te wijten aan het feit dat het met de hand moet gebeuren. Routine-rapporten en andere vormen van uitvoer worden meestal automatisch verkregen met de snelheid van de computer en kosten naar verhouding weinig tijd.

Hoe vaak de database bijgewerkt moet worden, hangt grotendeels af van uw behoeften. Gewoonlijk moet een deel van het bijwerken dagelijks gedaan worden. Andere dingen hoeven enkel wekelijks, maandelijks enzovoorts gedaan te worden. Ander bijwerken wordt alleen gedaan als het echt nodig is.

Ons slijterijtje bijvoorbeeld zou zijn inventarisdatabase kunnen bijwerken iedere keer dat een nieuwe zending binnenkomt. De tijdsbesteding van werknemers zou, afhankelijk van de situatie, dagelijks of wekelijks bijgewerkt kunnen worden. De omvang van het werk zal natuurlijk afhangen van de toepassing die u hebt.

Veranderingen kunnen meestal in een van de volgende rubrieken ingedeeld worden;

1 Het veranderen van de opzet van de database

2 Het toevoegen en verwijderen van records

3 Het veranderen van de inhoud van records

De opzet van de database veranderen

In het algemeen wordt er weinig aan de opzet van een database veranderd. De opzet wordt meestal veranderd, wanneer de omstandigheden veranderen, waaronder het bedrijf werkt – bijvoorbeeld door nieuwe voorschriften van de overheid. Omdat daarbij gegevens verloren kunnen gaan, moet u erg voorzichtig zijn.

Iedere keer dat een opzet wordt veranderd (bij dBASE II gaat dat met MODIFY STRUCTURE), zal de inhoud van de database worden gewist. Om hiervan geen last te hebben moet u steeds eerst een kopie maken van de database, voordat u de opzet gaat veranderen. Daarna kunt u de structuur veilig veranderen.

Een voorbeeld hiervan bij het dBASE II systeem is te zien in Scherm 6-1.

```
.USE B:INVENTRS
.COPY ALL TO B:NIEWFILE
00015 RECORDS COPIED
.MODIFY STRUCTURE
MODIFY ERASES ALL DATA RECORDS...PROCEED? (Y/N) Y
```

Scherm 6-1

Voorbeelden van veranderingen die in de structuur aangebracht kunnen worden, zijn:

Velden toevoegen
Velden verwijderen
De naam van een veld veranderen (speciale bewerking)
De breedte van een veld wijzigen
Het type van een veld veranderen

De computer zal de structuur van de database op de terminal laten zien als in Scherm 6-2.

```
NAME                TYPE   LEN  DEC
FIELD 01:DRANK      C      010  000  :
FIELD 02:MERK       C      020  000  :
FIELD 03:INHOUD     C      007  000  :
FIELD 04:VOORRAAD   N      003  000  :
FIELD 05:KOSTEN     N      006  002  :
FIELD 06:PRIJS      N      006  002  :
FIELD 07:                           :
FIELD 08:                           :
FIELD 09:                           :
FIELD 10:                           :
FIELD 11:                           :
FIELD 12:                           :
FIELD 13:                           :
FIELD 14:                           :
FIELD 15:                           :
FIELD 16:                           :
```

Scherm 6-2

In deze werkstand bent u in staat de cursor over het scherm te laten bewegen om de gewenste veranderingen aan te brengen. De cursor kan in sommige gevallen verplaatst worden door gebruikmaking van de cursorbesturingstoetsen. Als uw toetsenbord niet over die toetsen beschikt, kan de cursor verplaatst worden door gebruik te maken van de control toetsen aangegeven in Figuur 6-1. Een veld kan verwijderd worden door de cursor op de veldnaam te zetten en te drukken op hetzij CONTROL Y, hetzij CONTROL T. Een veld kan worden toegevoegd door de cursor op de plaats te zetten waar u het nieuwe veld wilt hebben en op CONTROL N te drukken. De velden erna zullen dan allemaal één plaats naar beneden opschuiven en op de plaats van het nieuwe veld komt een blanco regel te staan. Type de VELDNAAM, het VELDTYPE en de BREEDTE in en maak zo het toevoegen van het veld af. Al aanwezige velden kunnen gewijzigd worden door de cursor te verplaatsen naar het betreffende veld en de nieuwe informatie over de oude heen te typen. Het typen van CONTROL W maakt de computer duidelijk, dat u klaar bent met het wijzigen van de structuur. CONTROL Q breekt het wijzigen voortijdig af.

Controltoets	Uitwerking
S	Verplaatst de cursor 1 teken naar links
D	Verplaatst de cursor 1 teken naar rechts
E	Verplaatst de cursor 1 veld achteruit
X	Verplaatst de cursor 1 veld vooruit
T	Verwijdert het veld
Y	Wist de inhoud van het veld
N	Voegt een blanco veld in op de positie van de cursor
W	Schrijft de nieuwe filestructuur weg
Q	Annuleert het veranderen

Figuur 6-1 Werking van de controltoets bij MODIFY STRUCTURE

U hebt nu een kopie van de oude database en een nieuwe database zonder records. De gegevens uit de oude database, die thuishoren in de nieuwe database worden in de nieuwe database geladen met het dBASE II commando APPEND.

```
.APPEND FROM B:NIEWFILE
00015 RECORDS APPENDED
```

De inhoud van alle velden die de oude en de nieuwe database gemeen hebben, zal toegevoegd worden aan de nieuwe. Nieuwe velden blijven blanco. De gegevens in een veld dat minder breed gemaakt is, zullen afgebroken worden, wanneer het aan de nieuwe database wordt toegevoegd. Tekenvelden zullen hun meest rechtse tekens verliezen, terwijl getalvelden hun meest linkse cijfers zullen verliezen. Er ontstaat een probleem wanneer een veldnaam veranderd is. In dit geval zal de computer aannemen, dat dit veld een nieuw veld is en dat het oude verwijderd is. In het nieuwe record zal dit veld dus blanco zijn.

Bij het veranderen van een veldnaam moet u op een bijzondere manier tewerk gaan. Eén manier om een veldnaamverandering voor elkaar te krijgen is het 'uitladen' en vervolgens weer opnieuw 'laden' van de database. Wanneer u deze werkwijze volgt, mogen geen velden worden toegevoegd of worden verwijderd. Verder moeten alle veldbreedten onveranderd blijven. Bij dBASE II kan dit gedaan worden met een variant op het commando COPY. (Bij sommige systemen heet dit UNLOADING, wat een speciaal soort kopie maken is. De geUNLOADe database kan niet rechtstreeks door het database management systeem worden gebruikt. Deze speciale kopie wordt gebruikt om het database systeem te herladen als eenmaal de veranderingen in de opbouw zijn aangebracht.)

. COPY ALL TO B:NIEWFILE SDF

Hiermee wordt de inhoud van de database opgeslagen in NIEWFILE op een heel speciale manier. Er bestaan geen velden meer – hoewel de gegevens de vorm hebben van een stel rijen en kolommen. De file kan niet rechtstreeks gebruikt worden door het database management systeem. Hij kan echter wel gebruikt worden door een tekstverwerkingssysteem. SDF betekent System Data Format. Dit is een voorbeeld van een geUNLOADe database.

De 'uitgeladen' gegevens die tijdelijk zijn opgeslagen in de file B:NIEWFILE kunnen in de database INVENTRS geladen worden door

.APPEND FROM B:NIEWFILE SDF

Als de opbouw van INVENTRS dezelfde is als eerst, behalve dat er een veldnaam is veranderd – is de database nu weer in zijn oorspronkelijke toestand teruggebracht – maar dan met de nieuwe veldnaam.

Het is redelijk te verwachten dat de opbouw van de database niet vaak veranderd zal worden. Wanneer dat toch gebeurt, is het aan te bevelen een extra kopie te maken – voor het geval dat er iets mis gaat. Dit is namelijk een van die gevallen, waarin een fout wel erg zwaar gestrafd wordt.

RECORDS TOEVOEGEN EN VERWIJDEREN

Een veel gewoner soort verandering is het toevoegen of verwijderen van records. Nieuwe werknemers worden aangesteld, andere vertrekken of gaan met pensioen. Deze veranderingen zijn doorgaans heel eenvoudig.

Het toevoegen van een record aan een database is beschreven in de Hoofdstukken 2 en 5. Bij dBASE II gaat dit met APPEND. Telkens wanneer hiermee een record wordt toegevoegd, wordt het aan het einde van de database gezet. Dat einde heet bij dBASE II de onderkant. Er kunnen wel eens omstandigheden zijn, waarin het gewenst is een record tussen te voegen middenin de database. Bij dBASE II kan dat met het commando INSERT. Dit commando werkt net zo als APPEND, behalve dat u het record op iedere door u gewenste plaats in de database kunt tussenvoegen. Om dit commando te gebruiken moet u zich eerst verplaatsen naar een bepaald punt in de database. Dit doet u met het commando GOTO.

.GOTO RECORD 127

Bij gebruik van INSERT zal een blanco record worden tussengevoegd op recordplaats 128. Alle records die na record 127 kwamen, zullen een nieuw nummer krijgen en één plaats naar beneden gaan. Het oude record 128 wordt record 129, het oude record 129 wordt record 130 enzovoorts.

Zodra het hernummeren een feit is, zal het scherm gewist worden en het blanco record 128 zal u worden voorgezet voor het invoeren van gegevens als bij APPEND.

Records kunnen verwijderd worden in twee stappen. Eerst krijgt het record een merkje dat het verwijderd moet worden. Hier volgen enkele voorbeelden van commando's waarmee records gemerkt worden ter verwijdering:

.DELETE RECORD 216

.DELETE FOR NAAM = 'JANSEN, JAN'

.DELETE FOR DRANK = 'BIER'.AND.MERK = 'VABARIA'

Ieder keer dat een DELETE commando gegeven wordt, zal de computer antwoorden met het aantal records, dat verwijderd zal worden. Dit stelt u in de gelegenheid zonodig maatregelen te nemen om records terug te halen die per ongeluk gemerkt zijn. Het dBASE II commando om een weglating te 'herstellen' is RECALL. Hiervan volgen nu een paar voorbeelden:

.RECALL ALL

.RECALL RECORD 216

.RECALL FOR NAAM = 'JANSEN, JAN'

Het feitelijk weghalen van een of meer records gebeurt met een tweede commando. Bij dBASE II is dat PACK. PACK zal de records die u van een verwijderingsmerkje hebt laten voorzien door DELETE, voorgoed verwijderen.

Als een database wordt gebruikt samen met tabellen zoals dBASE II INDEX files, kan het nodig zijn de database iedere keer opnieuw van een index te voorzien, als records zijn toegevoegd of verwijderd. Sommige relationele database management systemen staan het toevoegen en verwijderen van records toe, terwijl de indexfile in gebruik is. Is dit het geval wordt de indexfile automatisch bijgewerkt bij het toevoegen of verwijderen van een record. Dit is meestal zo bij hiërarchische en netwerk (CODASYL) database systemen.

Wanneer een record wordt toegevoegd aan een database waar een indexfile bij is, zal dat record ingevoegd lijken in de database. Als de database geïndexeerd is op NAAM, zullen de records in alfabetische volgorde verschijnen. Een nieuw record zal geplaatst lijken te zijn op de juiste plaats in de alfabetische volgorde. Het wordt in feite toegevoegd aan het einde van de database, net als in het geval zonder indexfile.

DE INHOUD VAN RECORDS VERANDEREN

Er zijn verschillende manieren waarop u de inhoud van gegevensvelden kunt veranderen. Wanneer u bepaalde veranderingen aanbrengt in bepaalde records – bijvoorbeeld het invoeren van het aantal uren dat een werknemer op maandag gewerkt heeft – dan kan dat het best record voor record gebeuren op het hele scherm, zoals bij het dBASE II commando EDIT. Het EDIT commando geeft hetzelfde schermbeeld als het commando APPEND.

EDIT is het enige commando waarvoor u het recordnummer moet kennen. Veronderstel dat we in de inventarislijst van de slijterij het record willen wijzigen van de literflessen Andalucia sherry. Als we toevallig weten, dat het recordnummer van dit artikel 1 is, kunnen we het wijzigen laten beginnen met:

```
.EDIT 1
```

Het scherm zal eruit komen te zien als Scherm 6-3. Het record kan worden gewijzigd door de cursor te verplaatsen naar het gewenste veld en de nieuwe informatie in te typen. De cursor kan worden verplaatst met de cursorbesturingstoetsen. Als er geen cursorbesturingstoetsen zijn, kan de cursor verplaatst worden door gebruikmaking van de CONTROL toets, zoals te zien is in Figuur 6-2.

Om de mogelijkheden van EDIT te laten zien gaan we de VOORRAAD van 23 in 33 veranderen, de KOSTEN in 5.75 en de PRIJS in 10.98. Dit gaat als volgt in zijn werk. Houd de CONTROL toets ingedrukt. Terwijl u die ingedrukt houdt, slaat u driemaal de X aan. Daarmee verplaatst u de cursor van het veld DRANK naar het veld VOORRAAD. Type dan 33 in en een RETURN. De waarde 23 wordt zo vervangen door 33 en de cursor komt vooraan het veld KOSTEN te staan. Type 5.75 in en een RETURN. Zo vervangt u de waarde 5.59 door 5.75 en verplaatst u de cursor naar het veld PRIJS. Type 10.98 in en een RETURN. Dat vervangt de waarde 9.31 door 10.98 en u gaat door naar het volgende record – in dit geval Record 2. Druk op CONTROL R. Zo kiest u het vorige record – Record 1. Dat zal er nu uitzien als Scherm 6-4.

```
RECORD #00001

DRANK          :SHERRY      :
MERK           :ANDALUCIA               :
INHOUD         :1.0 LTR:
VOORRAAD       : 23:
KOSTEN         :  5.59:
PRIJS          :  9.31:
```

Scherm 6-3

Buurrecords kunnen gekozen worden met ^R (voor het vorige record) en ^C (voor het volgende record). Het bewerken kan afgebroken worden door gebruik te maken van ^Q. Het annuleren van de veranderingen kan heel handig zijn, als u ontdekt, dat u om een of andere reden het verkeerde record gewijzigd hebt. De veranderingen kunnen blijvend gemaakt worden met hetzij ^W, hetzij ^R, hetzij ^C. Het commando EDIT zal u in staat stellen de database record voor record te doorlopen door herhaald gebruik van ^C.

Controltoets	Uitwerking
S	Verplaatst de cursor 1 teken naar links
D	Verplaatst de cursor 1 teken naar rechts
E	Verplaatst de cursor 1 veld achteruit
X	Verplaatst de cursor 1 veld vooruit
Y	Wist de inhoud van het veld
U	Wissel – toevoegen/weglaten van record
V	Wissel – zet tussenvoegen van tekens aan/uit
W	Schrijft de nieuwe fileinformatie weg
Q	Annuleert het veranderen
R	Ga het vorige record wijzigen
C	Ga het volgende record wijzigen

Figuur 6-2 Werking van de controltoets bij EDIT

```
RECORD # 00001

DRANK          :SHERRY      :
MERK           :ANDALUCIA               :
INHOUD         :1.0 LTR:
VOORRAAD       : 33:
KOSTEN         :  5.75:
PRIJS          : 10.98:
```

Scherm 6-4

Wanneer het zou voorkomen – en meestal is dat zo – dat u het recordnummer niet kent, kunt u het op een aantal manieren krijgen, bijvoorbeeld:

```
.DISPLAY FOR DRANK = 'SHERRY'.AND.MERK = 'ANDALUCIA'

.LOCATE FOR DRANK = 'SHERRY'.AND.MERK = 'ANDALUCIA'.AND
.INHOUD = '1.0 L'
```

Op deze beide manieren krijgt u het recordnummer. Het tweede commando 'zet' de database bovendien op het gewenste record. In dit geval kan het record gewijzigd worden met EDIT. EDIT opent het wijzigen van het huidige record.

Een andere benadering die u in de gelegenheid stelt de inhoud van de database te bekijken en te wijzigen, is het dBASE II commando BROWSE. Een treffende beschrijving van BROWSE is het knippen van een vierkant gat uit een vel papier. Leg het vel met het gat erin nu over een ander, beschreven vel papier. U beschikt nu over een soort raam op dat tweede vel. Door het kijkgat te verschuiven kunt u beetje bij beetje de hele inhoud van het tweede vel bekijken. BROWSE is een raam dat zicht geeft op de database en u in de gelegenheid stelt steeds een stukje te bekijken en waar u dat wenst veranderingen aan te brengen. Het verschil tussen de functies BROWSE en EDIT is dat BROWSE u meerdere records tegelijk gedeeltelijk op het scherm zal laten zien, terwijl EDIT u één record in zijn geheel zal tonen.

De commando's BROWSE en EDIT kunt u slechts met de hand invoeren. Bovendien kan daarmee maar één record tegelijk veranderd worden.

Het databasesysteem moet ook zorgen voor de mogelijkheid meerdere records tegelijk te veranderen, wanneer een bepaalde omstandigheid gewijzigd is. Een voorbeeld hiervan is het vertrek van een bepaalde leerkracht en de komst van een andere. Als u EDIT of BROWSE gebruikt, moet u bij elk record dat veranderd moet worden, de naam van de leerkracht opnieuw typen. Een eenvoudiger oplossing in dit geval is het gebruik van het dBASE II commando REPLACE. Als de nieuwe leerkracht Mevrouw Jansen is en de oude leerkracht Meneer de Vries en het lokaalnummer 21, dan kan het commando een van de commando's hieronder zijn. Wanneer u REPLACE gebruikt met een voorwaarde, zullen alle records gewijzigd worden, die aan de voorwaarde voldoen.

.REPLACE LEERKRACHT WITH 'MW. JANSEN' FOR LOKAAL = '21'

.REPLACE LEERKRACHT WITH 'MW. JANSEN' FOR LEERKRACHT = 'HR. DE VRIES'

Bij databasetoepassingen waarbij vaak veranderingen aangebracht moeten worden, komt het soort procedures, dat u in het laatste hoofdstuk vindt, goed van pas: verduidelijkende prompts en MENU's. Het gebruik van procedures kan een goede vervanging zijn van uw geheugen. Ze maken het bovendien mogelijk de database te laten bewerken door minder deskundige (en dus goedkopere) krachten. In het laatste hoofdstuk zult u zien dat de twee manieren gecombineerd kunnen worden. Dat geeft een enorm gemak bij het bijwerken van uw database.

Scherm 6-5 toont hoe het veranderen van de database van de basisschool in zijn werk gaat bij gebruikmaking van een PROCEDURE. De procedure die gedemonstreerd wordt, gebruikt een combinatie van verduidelijkende prompts en menu's om degene die de gegevens invoert, te helpen. In dit voorbeeld worden records toegevoegd, records gewist en wordt de mogelijkheid geboden records te wijzigen. De records bevatten de NAAM, het LOKAAL, de KLAS, de LEERKRACHT en nog meer informatie over de leerling.

```
KIES EEN VAN DE VOLGENDE DINGEN

N – NIEUWE LEERLING TOEVOEGEN

V – VERANDEREN VAN INFORMATIE LEERLING

D – VERANDEREN VAN INFORMATIE VOLGENDE LEERLING

W – WISSEN VAN INFORMATIE LEERLING

Q – EINDE WERKEN MET COMPUTER

VOER AUB UW KEUZE IN  :  :
```

Scherm 6-5

Als V gekozen wordt, zal de computer een nieuw scherbeeld geven. Dit is afgebeeld in Scherm 6-6.

```
VOER AUB DE VOLGENDE INFORMATIE IN

KLAS VAN LEERLING  :  :

LOKAAL VAN LEERLING  :      :

ACHTERNAAM VAN LEERLING  :                    :

VOORNAAM VAN LEERLING  :                      :
```

Scherm 6-6

Wanneer de computer die informatie ontvangen heeft, zal hij even tijd nodig hebben om de gezochte leerling op te sporen. Vervolgens zal de computer het record van de leerling tonen als afgebeeld in Scherm 6-7.

GEGEVENS LEERLING

NAAM LEERLING (EERST DE ACHTERNAAM):**BAKKER, INEKE** :

LOKAAL :**201**: KLAS :**5**: NAAM LEERKRACHT :**JANSEN** :

GEBOORTEDATUM (DAG/MAAND/JAAR) :**27/11/71**:

.....en dan volgt de rest van het record van de leerling.

Scherm 6-7

De informatie die veranderd moet worden, wordt gekozen door de cursor naar het te veranderen veld te verplaatsen en de nieuwe informatie te typen. Wanneer de veranderingen zijn aangebracht, keert de gebruiker terug naar het eerste, het hoofdmenu. Een nieuwe verandering wordt gekozen en de bewerking wordt herhaald. Bij de menukeuze N (Nieuwe leerling) wordt rechtstreeks naar het record van de leerling toegegaan, dat natuurlijk helemaal blanco is. Met de keuze D (Doorgaan) kunt u van record naar record gaan, zoals wanneer u bijvoorbeeld proefwerkcijfers invoert. U hoeft dan geen dingen te typen, die voor dat doel niet nodig zijn. Bij menukeuze W (Wissen) zal het tweede schermbeeld nadere informatie vragen over de leerling. Wanneer wissen uitgekozen is, behoort de procedure altijd de gebruiker een tweede kans te geven. Na het invoeren van de nadere informatie moet de computer de gebruiker dus iets laten zien als het schermbeeld in Scherm 6-8.

INFORMATIE WISSEN

U GAAT DE VOLGENDE LEERLING AFVOEREN

BAKKER, INEKE

KLAS 5 LOKAAL 201

WEET U ZEKER DAT DEZE LEERLING WEGGELATEN MOET WORDEN (Y/N): :

Scherm 6-8

Dit is een goed voorbeeld van de toepassing van menu's en verduidelijkende aansporingen als hulp voor de persoon die de informatie in de computer moet bijhouden. Het lijkt een beetje op het gebruik van voorgedrukte FORMULIEREN als hulpmiddel bij de administratie. In dat geval zijn er geschreven voorschriften, die de medewerker vertellen, welk formulier gebruikt moet worden. Bij de computer is het de procedure die de computer vertelt, welk 'formulier' gebruikt moet worden. Als degene die achter het toetsenbord zit, niet vertrouwd is met de computer, kan de procedure ook nog een keuzemogelijkheid in het menu hebben, waarmee hulp kan worden ingeroepen.

Een van de bijkomstige voordelen van de procedure is het formaliseren van het veranderen van de database. Daarbij wordt tegelijk de moeite bij het bedienen van de computer tot een minimum teruggebracht. Als het invoeren van de gegevens interessant, zo niet echt boeiend, gemaakt kan worden, loopt de kans op fouten terug. Bij de meeste databasetoepassingen is het essentieel, dat de database geen fouten bevat. Bij sommige bewerkingen met behulp van menu's wordt het werk tot een minimum teruggebracht evenals de kans op spelfouten (bedenkt u, dat terwijl u zelf wel kunt weten, dat shery sherry betekent, de computer dat niet kan). De mogelijkheid bestaat nu wel, dat JENEVER wordt ingevoerd, in plaats van SHERRY. De procedure kan voorzorgsmaatregelen nemen tegen dergelijke vergissingen, maar dat vraagt dan wel een extra inspanning van degene die de procedure schrijft.

Het laatste onderwerp van dit hoofdstuk is het onderhouden van de database. In dit geval betekent 'onderhouden' 'veiligstellen' van de database. Als u een 'database' hebt, die bestaat uit een stelletje aantekeningen op papier, kunt u moeilijkheden verwachten bij het veiligstellen van de informatie op papier. Er kan koffie op gemorst worden of iemand kan per ongeluk een belangrijk velletje papier weggooien.

Voor die 'database' zijn echter maar weinig dingen op echte rampen na (brand, overstroming, windhoos enzovoorts) werkelijk gevaarlijk. Bij een computerdatabase is dat anders. Die staat op een flinterdun laagje gemagnetiseerd mylar. Een vinger die de mylar film per ongeluk aanraakt, kan de opgeslagen informatie al beschadigen. Een sigaret zal de database volledig vernietigen. Een klein stukje mylar film is kwetsbaarder dan een archiefkast vol papier. Alle dingen die het gemakkelijk maken de database te gebruiken, maken hem ook gevoelig voor onmiddelijke rampen.

Veel van bovenstaande ellende kan worden vermeden als u zich strikt houdt aan een paar eenvoudige vaste werkwijzen. Er moet gezorgd worden voor een of twee reserve exemplaren (backups) van de database, die ook bijgewerkt worden. Dit is een heel goede reden om tenminste twee diskettedrives bij uw computersysteem te hebben. Als u meer dan twee diskettedrives hebt, kunt u de kopie gemakkelijk maken door een lege diskette in een van de extra drives te doen en gebruik te maken van het commando COPY. Als de database in diskettedrive B zit en de lege diskette in drive C, dan wordt de kopie gemaakt door:

.COPY ALL TO C:BACKUP

Als u maar twee diskettedrives hebt, uw database zich in drive B bevindt en er genoeg ruimte beschikbaar is op de diskette in drive A om ook nog plaats te bieden aan de database, kan de kopie worden gemaakt met:

.COPY ALL TO A:BACKUP

Als u maar twee diskettedrives hebt, uw database zich in drive B bevindt en er niet genoeg ruimte beschikbaar is in drive A, gebruikt u dan het dBASE II commando QUIT. QUIT brengt u terug bij het besturingssysteem van uw computer.

.QUIT

*** END RUN DBASE II ***

A>◆

Als zich in drive A een exemplaar bevindt van CP/M, uw besturingssysteem, zal de volgende werkwijze u een backup exemplaar verschaffen van uw database. Haal de diskette met uw database uit diskettedrive B. Doe een lege diskette in drive B. Type ^C. Type de letters PIP achter de A>.

De computer zal antwoorden met een '*' op de regel onder de A>. Haal de diskette uit drive A. Doe de diskette met de database in drive A. Type B:=*.*[V]. Zo zal een kopie van de inhoud van diskettedrive A in diskettedrive B gezet worden, waarbij tevens controle plaatsvindt. Haal de diskette met de database en het backup exemplaar uit de drives. Breng de oorspronkelijke diskette in drive A terug. Doe de oorspronkelijke database terug in drive B. Druk op de RETURN toets. Het hele gebeuren is afgebeeld in Scherm 6-9.

```
.QUIT

*** END RUN          DBASE II ***

                     .....(haal databasediskette uit drive B)
                     .....(doe backupdiskette in drive B)

A>^C

A>PIP

                     .....(haal CP/M diskette uit drive A)
                     .....(doe databasediskette in drive A)

*B:=*.*[V]
*

                     .....(haal databasediskette uit drive A)
                     .....(doe CP/M diskette in drive A)

A>^C

A>◆

                     .....(u hebt nu een backupdiskette in B)
```

Scherm 6-9

Eén benadering voor het veilig bewaren van uw files is het hebben van twee exemplaren. De twee exemplaren worden om de andere dag gebruikt. Het gebruik van twee backup exemplaren zal op de lange duur in uw voordeel zijn. Goed toegepast beschermt deze werkwijze u tegen alle verlies van werk, behalve misschien dat van één dag. Als u het risico hiertoe kunt terugbrengen (onder uitsluiting van een echte ramp, brand en dergelijke), hebt u zo ongeveer alles gedaan, wat in uw vermogen ligt. Het is zelfs mogelijk een betere bescherming te hebben tegen grote rampen dan mogelijk is bij een archief op papier. U kunt niet twee exemplaren van een archief op papier op twee verschillende plaatsen bijhouden. Bij een computerdatabase kan dat echter best. Omdat u gemakkelijk kopiëen van de database op diskettes of op cassettes zet, kunnen de kopieën op een heel andere plaats bewaard worden, waarmee u zo ongeveer beveiligd bent tegen alle onheil behalve oorlog.

7. DE DATABASE GEBRUIKEN

Aan het omgaan met uw database zitten twee kanten. De ene kant, het bijwerken en onderhouden ervan, is behandeld in het vorige hoofdstuk. Het tweede aspect, de database iets voor u laten doen, is het onderwerp van dit hoofdstuk.

DE TWEE HOOFDTOEPASSINGEN VAN DE DATABASE

De database kan in hoofdzaak op twee manieren gebruikt worden:

1 Om standaarddingen te doen, bijvoorbeeld salarisoverzichten maken, belastingaangiftes, voorraadlijsten enzovoorts.

2 Om wanneer nodig bepaalde informatie te verkrijgen.

Ons database management systeem, dBASE II, heeft een rapportenschrijver, bedoeld voor het maken van allerlei standaardrapporten. Daarnaast kunt u ook 'procedures' schrijven, om speciale rapporten op maat te maken, afgestemd op uw eigen behoeften. Bijzondere informatie die nodig is voor dingen die geen routine zijn, wordt via het toetsenbord van de computer ingebracht door gebruikmaking van een 'vraagtaal'. We zullen het eerst hebben over 'rapporten', zoals we ze in Hoofdstuk 2 al even zijn tegengekomen. Verderop in dit hoofdstuk hebben we het over vraagtalen en – processen.

RAPPORTEN

In Hoofdstuk 1 hebben we het voorbeeld van de inventarislijst van de slijterij gebruikt om u het schrijven van rapporten te laten zien. De meeste moderne database systemen beschikken over een voorziening om snel rapporten te maken met informatie uit de database. De standaardmogelijkheden van het rapport kunnen vanaf het toetsenbord worden verkregen, waarbij u als gebruiker wordt voorzien van informatie die komt uit de database. In Hoofdstuk 2 hebben we gezien, dat een eenvoudige dialoog een raamwerk voor een rapport opzet, dat de computer zich kan herinneren. We kunnen een dergelijk rapportformulier telkens opnieuw gebruiken. U maakt hiermee een begin via:

```
.REPORT
```

De dialoog die hierop volgt, is voor uw gemak herhaald uit Hoofdstuk 2 in Scherm 7-1. Ernaast zijn hier nog wat extra opmerkingen in kleine letters toegevoegd. Het rapport dat de gebruiker samenstelt uitgaande van de database B:INVENTRS via het rapportageformulier ingevuld in Scherm 7-1, is afgebeeld als Figuur 7-1. Als u wilt dat dit rapport op uw printer wordt afgedrukt, dan is het commando

.REPORT TO PRINT

```
ENTER REPORT FORM NAME: B:INVENTRS
ENTER OPTIONS, M = LEFT MARGIN, L = LINE/PAGE, W = PAGE
WIDTH
PAGE HEADING? (Y/N) Y
ENTER PAGE HEADING: INVENTARISLIJST SLIJTERIJ
DOUBLE SPACE REPORT? (Y/N) N
TOTALS REQUIRED IN REPORT? Y
SUBTOTALS IN REPORT? (Y/N) Y
ENTER SUBTOTALS FIELD:DRANK          (subtotalen voor elke drank)
SUMMARY REPORT ONLY? (Y/N) N         (geef ook alle bestanddelen)
EJECT PAGE AFTER SUBTOTALS (Y/N) N
ENTER SUBTOTAL HEADING:
COL      WIDTH, CONTENTS
001      20, MERK                    (aantal posities, veldnaam)
ENTER HEADING: MERK
002      7, INHOUD
ENTER HEADING: INHOUD
003      3, VOORRAAD
ENTER HEADING: VRD
ARE SUBTOTALS REQUIRED? (Y/N) Y
004      6, KOSTEN
ENTER HEADING: KOSTEN
ARE TOTALS REQUIRED? (Y/N) N         (* betekent vermenigvuldigen)
005      7, KOSTEN*VOORRAAD            (kosten maal hoeveelheid)
ENTER HEADING: INVEST
ARE TOTALS REQUIRED? (Y/N) Y
006
```

Scherm 7-1

Page No. 00001
09/15/81

Inventarislijst slijterij

Merk	Inhoud	Vrd	Kosten	Invest
* SHERRY				
ANDALUCIA	1.0 L	23	5.59	128.57
ANDALUCIA	2.0 L	7	9.78	68.46
ANDALUCIA	0.5 L	88	2.74	241.12
** SUBTOTAL **		118		438.15
* WODKA RUSSKI	1.0 L	35	3.78	132.30
RUSSKI	2.0 L	9	7.95	71.55
RUSSKI	0.5 L	75	1.4	111.75
** SUBTOTAL **		119		315.60
* WHISKEY				
SOUTHERN RYE	1.0 L	32	5.11	163.52
OLD IRISH	0.5 L	44	1.98	87.12
OLD IRISH	1.0 L	19	5.29	00.51
THE NEW SOUTH	1.0 L	4	7.49	29.96
** SUBTOTAL **		99		381.11
* JENEVER				
SCHIEDAM	0.5 L	5	0.99	4.95
SCHIEDAM	0.8 L	22	1.78	39.16
SCHIEDAM	1.0 L	21	3.50	73.50
SCHIEDAM	2.5 L	3	6.89	20.67
SCHIEDAM	2.0 L	5	6.47	32.35
* SUBTOTAL **		56		170.63
** TOTAL **		392		1305.49

Figuur 7-1 Computerrapport van inventarislijst slijterij

Dit rapport is duidelijk, omdat de 'soorten' drank in de database bij elkaar staan. Als ze niet bij elkaar stonden, zou het resutaat rommelig zijn. Wanneer we niet zeker weten, of de dranken wel of niet bij elkaar staan, behoren we eerst de database te indexeren om de gegevens te groeperen. Dat geeft dan een samenhangend rapport.

```
.INDEX ON DRANK TO B:DRANK
.USE B:INVENTRS INDEX B:DRANK
```

Een van de prompts voor het rapport, die we met nee beantwoord hebben in Scherm 7-1, was 'SUMMARY REPORT ONLY'. Bij een 'Y' voor 'ja' op deze prompt krijgen we een ander rapport dan dat in Figuur 7-1. Een dergelijk verkort rapport is hieronder te zien in Figuur 7-2. Merkt u op, dat er hier geen informatie staat in de kolom kosten. De enige waarden die afgedrukt zijn, zijn de subtotalen per rubriek.

Page No. 00001
09/15/81

Inventarislijst slijterij

Merk	**Vrd**	**Kosten**
INVEST		
* SHERRY	118	438.15
* WODKA	119	315.60
* WHISKEY	99	381.11
* JENEVER	56	170.63
** TOTAL **	392	1305.49

Figuur 7-2 Door computer vervaardigd verkort overzicht van inventarislijst slijterij

Het ontbreken van informatie in de kolom kosten is een rechtstreeks gevolg van ons 'nee' op de aansporing 'SUBTOTALS IN REPORT?'

Gebruik makend van het oorspronkelijke voorbeeld, vragen we een gedetailleerd rapport over de jenever:

```
.REPORT FOR DRANK = 'JENEVER'
```

Hiermee krijgen we het rapport afgebeeld in Figuur 7-3. Dit rapport geeft ons informatie over jenever alleen. Omdat onze database zo klein is, heeft dit voorbeeld alleen waarde als illustratie. U ziet de mogelijkheid een rapport te laten uitbrengen over iedere aanwijsbare deelverzameling van de database. Deze voorziening is van onschatbare waarde bij een grote database. Als hij groot is, kunt u de afdruk eerst in verkorte vorm laten maken. Daarna worden volgende rapporten, zoals Figuur 7-3, gekozen op grond van de informatie in dat beknopte rapport.

Page No. 00001
09/15/81

Inventarislijst slijterij

Merk	Inhoud	Vrd	Kosten	Invest
* JENEVER				
SCHIEDAM	0.5 L	5	0.99	4.95
SCHIEDAM	0.8 L	22	1.78	39.16
SCHIEDAM	1.0 L	21	3.50	73.50
SCHIEDAM	2.5 L	3	6.89	20.67
SCHIEDAM	2.0 L	5	6.47	32.35
* SUBTOTAL **		56		170.63
** TOTAL **		392		1305.49

Figuur 7-3 Computerrapport van de jenever in de inventarislijst van de slijterij

Als we een nog sneller, maar wel wat minder elegant resultaat willen, kunnen we twee commando's uit de Vraagtaal gebruiken. Over vraagtalen en vragen gaan we het zo meteen hebben, maar we nemen dit voorbeeld hier op om de twee manieren waarop u uw database kunt gebruiken, naast elkaar te zetten. De Vraagtaalcommando's en antwoorden van de tweede manier zijn te zien in Scherm 7-4.

```
.DISPLAY OFF MERK, VOORRAAD, KOSTEN, KOSTEN*VOORRAAD
FOR DRANK = 'JENEVER'

SCHIEDAM        0.5 L     5      0.99        4.95
SCHIEDAM        0.8 L    22      1.78       39.16
SCHIEDAM        1.0 L    21      3.50       73.50
SCHIEDAM        2.5 L     3      6.89       20.67
SCHIEDAM        2.0 L     5      6.47       32.35

.SUM VOORRAAD, KOSTEN*VOORRAAD

                         392                1305.49
```

Scherm 7-4

U kunt uit ons nogal uitgebreide gebruik van het voorbeeld van de inventarislijst van de slijterij zien, dat het onttrekken van informatie aan een database via een rapport echt heel eenvoudig is. In het bedrijf is het niet alleen gemakkelijk, maar ook relevant om een groot aantal speciale toepassingensrapporten af te leiden uit uw database. Dat kunnen crediteuren- en debiteurenoverzichten zijn, salarisafrekeningen, personeelslijsten enzovoorts, en ook een voorraadbeheerssysteem als het onze voor de slijterij. Er schuilt veel voordeel in het nemen van één database als uitgangspunt voor al deze toepassingen. Elke afzonderlijke toepassing kan gemakkelijk zijn eigen procedure of zijn eigen rapport op maat krijgen. Tegelijkertijd echter komt alles uit een gemeenschappelijke database – een gemeenschappelijke informatiebron die toegankelijk is voor bepaalde niet-routinematige activiteiten naast het standaardgebruik dat routinematig op gezette tijden plaatsheeft.

Een voorbeeld. Stel dat bij een bedrijf over een nieuwe CAO wordt onderhandeld. De bedrijfsleiding wil graag de gevolgen kennen van de overeenkomst, voordat men met de voorwaarden akkoord kan gaan. Als de personeels- en salarissystemen van elkaar gescheiden zijn – zoals vaak het geval is, zelfs in grote, professioneel ontworpen en geleide organisaties – kan het heel wat werk kosten de gevolgen van de overeenkomst te bepalen, zo het al mogelijk is.

Als personeel en salarissen gescheiden systemen zijn, wordt veel werk dubbel gedaan – en moet veel informatie dubbel worden ingevoerd. Een computersysteem voor de salarisadministratie zal de naam van de werknemer nodig hebben, zijn of haar nummer, gezinssamenstelling, salarisschaal, enzovoorts. Een personeelssysteem heeft dergelijke informatie ook nodig. Als de systemen gescheiden zijn, kan de informatie die in het ene systeem is opgeslagen, niet altijd door het andere gebruikt worden.

Als daarentegen zowel het personeelssysteem als het salarissysteem één database systeem gebruiken, kan de informatie rechtstreeks vanaf het toetsenbord toegankelijk zijn via een eenvoudige vraag.

Deze 'scheiding' tussen gegevens die vaak onvermijdelijk met elkaar verbonden zijn, was een van de factoren die geleid hebben tot de ontwikkeling van databases. Bij een database is in de computer de opgeslagen informatie onafhankelijk van het toepassingsgebied, zulks in tegenstelling tot bijvoorbeeld een 'crediteurenprogramma'. Het is aan de gebruiker de toepassing aan te geven, die hij of zij wil zien, wanneer hij of zij de database aanspreekt. Dit brengt ons op iets dat al een paar keer genoemd is: informatie beschikbaar via vragen.

INFORMATIE OPVRAGEN (QUERY)

Database systemen geven antwoord op vragen van de gebruiker om informatie (queries). Het deel van het database systeem, dat dit doet, heet de QUERY LANGUAGE PROCESSOR (QLP). De vraagtaalverwerker van dBASE II heet APPLICATIONS DEVELOPMENT LANGUAGE (ADL).

U gebruikt de VRAAGTAAL om de computer duidelijk te maken, wat hij moet doen. De meeste VRAAGTALEN van tegenwoordig lijken heel erg op gewoon Engels. Bij bepaalde systemen is de enige taak van de QLP het opvragen van informatie uit de database. In dit geval noemt men de QLP een 'READ-ONLY' (alleen lezen) functie. Bij dBASE II wordt ADL ook gebruikt om de database bij te werken. Dit betekent, dat de taal zowel kan 'SCHRIJVEN' als kan 'LEZEN'.

VRAAGTALEN

Er bestaan twee soorten vraagtalen: procedurele en niet-procedurele.

- Een procedurele taal is de klassieke computertaal. Met dit soort taal vertelt u de computer, stap voor stap, wat u hem wilt laten doen om het door u gewenste resultaat te krijgen. Voorbeelden hiervan zijn BASIC, FORTRAN, PL/1 en COBOL. Bij al deze talen vertelt u de computer, hoe hij het antwoord moet vinden – niet wat het probleem is.
- Niet-procedurele talen stellen u in staat het probleem te formuleren, waarna de computer uitzoekt, hoe hij aan het antwoord moet komen.

Uit het voorbeeld van het persoonlijke telefoonlijstje is

```
.COUNT FOR 'Arnhem'$ADRES
```

een voorbeeld van een niet-procedureel commando, waarmee u te weten kunt komen, hoeveel vermeldingen in Arnhem er zijn. U hebt de computer verteld, wat u wilt en de computer zoekt zelf uit hoe hij erachter komt.

Bepaalde vraagtalen hebben eigenschappen van beide soorten. De ADL van dBASE II is zowel een procedurele als een niet-procedurele taal. De procedurele en niet-procedurele voorzieningen van vraagtalen voor relationele database systemen worden soms met technische termen aangeduid:

- Procedurele voorzieningen heten de RELATIONELE ALGEBRA.
- Niet-procedurele voorzieningen heten de RELATIONELE ANALYSE.

Het is niet duidelijk, of deze termen nog een andere betekenis hebben dan een poging tot intimidatie van de buitenstaander te zijn.

Bepaalde informatie die nodig is in niet-routine gevallen, wordt verkregen vanaf het toetsenbord van de computer door gebruik te maken van de Vraagtaal. Als de database een groot aantal velden heeft, moet u hiervoor wellicht een data dictionary bijhouden en gebruiken (Hoofdstuk 3). Om de database namelijk onder alle omstandigheden te kunnen gebruiken moet u de VELDNAMEN kennen en weten, wat de VELDEN bevatten.

Een vraagtaal die u in de gelegenheid stelt op hoog niveau vragen te stellen vanaf het toetsenbord, kent commando's, die uit drie delen bestaan. Dat zijn:

- de COMMANDONAAM
- het BEREIK en
- de VOORWAARDE.

De 'COMMANDONAAM' geeft gewoonlijk aan, wat de functie doet. Voorbeelden van commandonamen bij dBASE II zijn DISPLAY, SUM, COUNT, LOCATE en LIST. Het bereik geeft aan, op welk deel van de database het commando betrekking heeft.

De voorwaarde houdt in, dat het commando van toepassing is op die databaserecords die voldoen aan de vermelde voorwaarde. Voorbeelden van mogelijke vragen zijn:

```
.SUM VOORRAAD FOR DRANK = 'JENEVER'

.COUNT FOR 'Rob'$NAAM

.DISPLAY VOORRAAD FOR DRANK = 'SHERRY'.AND.INHOUD = '0.8 L'

.DISPLAY VOORRAAD, MERK FOR DRANK = 'SHERRY'.AND.INHOUD = '0.8 L'
```

In het eerste van de vier voorbeelden hierboven betekent het commando 'vertel ons hoeveel flessen jenever we hebben'. Meer uitgewerkt, betekent het 'tel de inhoud van het veld VOORRAAD op voor alle records waarbij het veld DRANK 'JENEVER' bevat'. Als we VOORRAAD uit het commando weglaten, verliest het zijn betekenis. U kunt geen onduidelijke opdracht aan de computer aanbieden. Als u dat doet, zal het antwoord evenmin betekenis hebben. Bij een goede vraagtaal zal de computer het commando weigeren, als u zoiets probeert.

In het tweede voorbeeld betekent het commando 'tel de records die de naam Rob bevatten'. Het derde voorbeeld vraagt (enkel) de voorraad te tonen van alle soorten sherry in flessen van 0.8 liter. In het vierde voorbeeld wordt niet alleen de voorraad, maar ook het merk gevraagd.

Al deze voorbeelden lijken opmerkelijk veel op gewoon Engels. Dat komt ten dele omdat we alle velden een goede omschrijving hebben gegeven als veldnaam.

Bij ADL wordt door het commando DISPLAY gewoonlijk het hele record getoond. Als u minder wenst dan dat, maakt het invoeren van een lijst door komma's gescheiden veldnamen (als in het vierde voorbeeld) de computer duidelijk, dat alleen de genoemde velden weergegeven moeten worden.

Het vierde voorbeeld gebruikt het woord AND op een beetje vreemde manier. We hebben hier tegen de computer gezegd: als de inhoud van het veld DRANK 'SHERRY' is en de inhoud van het veld INHOUD '0.8 L' is, toon dan het merk en de hoeveelheid die ervan in voorraad is. Deze vraag zal precies dat wat we wensen, uit de database halen.

Stel dat we de volgende vraag invoeren:

.DISPLAY FOR DRANK = 'SHERRY'.AND.DRANK = 'JENEVER'

We willen dat de computer alle records weergeeft, waarbij de DRANK SHERRY is en die waarbij de DRANK JENEVER is. Het antwoord van de computer op dit commando is een puntaansporing. Er wordt niets getoond. Dit betekent dat de computer niets gevonden heeft. Hoe kan dat nou? We weten dat er zowel sherry- als jeneverecords zijn ingevoerd en toch zegt de computer dat er geen zijn. Het antwoord is, dat dat 'geen' nog waar is ook; er is geen record, waarvan de drank zowel sherry als jenever is. Als we de vraagzin als volgt herschrijven: 'We willen dat de computer alle records weergeeft, waarin de soort drank hetzij sherry, hetzij jenever is' – is het duidelijk. We denken aan de hele database wanneer we om informatie vragen. De computer werkt met één record tegelijk. Als we het commando herschrijven tot:

.DISPLAY FOR DRANK = 'SHERRY'.OR.DRANK = 'JENEVER'

krijgen we het gewenste antwoord.

LOGISCHE OPERATOREN

De AND en de OR heten logische operatoren. De LOGISCHE operatoren worden ook wel BOOLEse operatoren genoemd. Zoals we gezien hebben zijn de .AND. en de .OR. bijna hetzelfde als in gewoon Engels. Wanneer we deze operatoren gebruiken, worden ze met punten aan weerszijden geschreven om ze te kunnen onderscheiden van hun evenbeelden in gewoon Engels. U moet erg voorzichtig zijn om vreemde resultaten te voorkomen. Het is van groot belang dat u wat moeite doet en ervoor zorgt, dat u de logische operatoren begrijpt. Anders kunt u een TECHNISCH JUIST maar niettemin FOUT antwoord krijgen. Dit is een geval waar de computer precies doet, wat u hem vertelt te doen – niet wat u misschien had gewild dat hij deed. Nog een logische operator die bij dBASE II gebruikt wordt, is .NOT., die nader wordt beschreven in Hoofdstuk 10.

MEER OVER VRAGEN

Vragen sluiten aan op een grote verscheidenheid aan behoeften bij de gebruiker. Misschien willen we – om een of andere reden – gegevens in de database bewerken, maar niet echt de gegevens in de 'officiële' database veranderen. We hebben hiervan al een voorbeeld gezien, toen we een kopie maakten van een database om de opbouw te veranderen. Het is dikwijls nuttig een kopie van een deel van een database in een nieuwe database te zetten.

Laten we eens een kopie maken van dat deel van B:INVENTRS dat bestaat uit de velden MERK, INHOUD en PRIJS en meer niet. Om de nieuwe database verder te beperken, willen we in de kopie alleen de records met 'sherry' hebben. Dit kunt u met één enkel commando doen.

```
.COPY FIELD MERK, INHOUD, PRIJS TO B:SHERRY FOR DRANK = 'SHERRY'
```

Hiermee maakt u een database van 'enkel sherry'. Om nog eens wat uit te weiden over de terminologie van specialisten: dit soort kopiëren heet 'het projecteren van de relatie B:INVENTRS in B:SHERRY met als beperking het predikaat DRANK='SHERRY'. Zou u die beschijving van ons simpele kopietje herkennen?

Laten we nog eens een 'willekeurige' wens bedenken. Stel dat we de dringende behoefte hebben te weten, van welk artikel in de slijterij er het meest is, en van welk het minst. Een index levert een lijst van de records geordend op voorraad.

```
.INDEX ON VOORRAAD TO B:VRD
00015 RECORDS INDEXED
.USE B:INVENTRS INDEX B:VRD
```

Om naar het begin te komen van de database met index gebruiken we het dBASE II commando GO TOP. U kunt naar het einde van de database gaan met GO BOTTOM.

```
.GO TOP
.DISPLAY OFF MERK, INHOUD, VOORRAAD
SCHIEDAM                  2.5 L          3
.GO BOTTOM
.DISPLAY OFF MERK, INHOUD, VOORRAAD
ANDALUCIA                 0.5 L         88
```

Via het toetsenbord hebben we onze vraag heel rechtstreeks beantwoord gekregen. Het artikel dat we het minst in voorraad hebben, is 2.5 literflessen Schiedam en het artikel waarvan we het meest in voorraad hebben, is halve-liter flessen Andalucia.

Deze vragen zijn een heel klein aselect steekproefje van dingen die u uw database kunt vragen. De mogelijkheden zijn in feite onbeperkt. Het gaat om het idee en om voldoende begrip van het stellen van zinnige vragen die bruikbaar zijn, bezien in het licht van uw eigen behoeften.

UW ELECTRONISCH KLADBLOK

Nog iets nuttigs staat tot uw beschikking: een soort kleine toevoeging aan alle grote mogelijkheden van uw database en uw computer.

Gegevens uit een afzonderlijk veld in een record kunnen voor uw gemak naar een speciale plaats in het geheugen van de computer gebracht worden. Voor uw gemak ook, kunt u de computer iets wat u via het toetsenbord invoert, laten opslaan in zijn hoofdgeheugen. Het is net alsof we een 'kladblok' tot onze beschikking hebben.

Een database systeem op een microcomputer stelt deze mogelijkheid ter beschikking in zijn Query Language Processor. De mogelijkheid van tijdelijke opslag komt al van pas, wanneer u op het toetsenbord aan het typen bent. Deze voorziening is echter van onschatbare waarde, wanneer u procedures aan het schrijven bent voor automatische bewerkingen.

De werking van het kladblok lijkt heel erg op het geheugen bij een elektronische rekenmachine. Bij dBASE II kunt u:

- tot 64 afzonderlijke variabelen opslaan in het kladblokgeheugen,
- met maximaal 254 bytes (tekens of cijfers) voor elke afzonderlijk opgeslagen variabele;
- en 1536 bytes totale opslagruimte voor alle 64 variabelen tezamen.

Iedere variabele die in het geheugen wordt opgeslagen, is een GEHEUGENVARIABELE. Wanneer iets (een GEHEUGENVARIABELE) in het geheugen is opgeslagen, moet er een naam aan toegekend worden. Die naam maakt het ons gemakkelijk de variabele terug te vinden en te gebruiken. Elk systeem gebruikt een of ander trefwoord om de computer opdracht te geven iets als geheugenvariabele op te nemen. Bij dBASE II is dat trefwoord STORE.

Om toe te lichten, hoe het 'kladblokgeheugen' werkt, gaan we het getal 6 opslaan in een geheugenvariabele die we VOORBEELD noemen.

.STORE 6 to VOORBEELD

Denkt u eraan:

- Met zekere beperkingen kunt u 64 afzonderlijke variabelen tegelijk in het geheugen opslaan.
  - Iedere keer dat u iets opslaat, moet u het een naam geven.
  - Als u dezelfde naam toewijst aan twee verschillende variabelen, zal alleen de als laatste ingevoerde variabele opgeslagen worden.
- Wilt u hetgeen in een geheugenvariabele is opgeslagen, veranderen, slaat u dan de nieuwe inhoud op onder de oude naam, .STORE 7 TO VOORBEELD zal bijvoorbeeld 7 in VOORBEELD stoppen in plaats van de 6, waarmee we hiervoor begonnen.
- U kunt geen twee variabelen tegelijk hebben met dezelfde naam.

Geheugenvariabelen kunnen tekenketens zijn, getallen of logische waarden.

.STORE 'ALFABET' TO SOEP

.STORE T TO ANTWOORD

U mag het geheugen bijna overal voor gebruiken. U kunt tekenketens samenvoegen tot een zin:

.STORE 'ALFABET' TO A

.STORE 'SOEP' TO B

.STORE A + B TO SOEP

Om te zien, wat zich in een geheugenvariabele bevindt, typen we een vraagteken achter de puntaansporing, gevolgd door de naam van de variabele.

.? SOEP
ALFABETSOEP

Met geheugenvariabelen kunt u rekenen.

```
.STORE 6 TO X
6
.STORE 7 TO Y
7
.STORE X+Y TO Z
13
.STORE Y-X TO W
1
.STORE X*Y TO Z          (* betekent vermenigvuldigen)
42
.STORE X/Y TO D          (/ betekent delen)
0                          (hier hebben een voorbeeld van de verdor-
                           venheid van computers. Om de cijfers achter
                           de komma te zien te krijgen, moeten we de
                           computer vertellen hoeveel we er wensen.)
.STORE X*1.0/Y TO D
0.8
.STORE X*1.000/Y TO D
0.857                      (snapt u het?)
```

Om u een beter idee te geven van het nut van het kladblokgeheugen gaan we eens kijken naar een overzicht in de computer van uitgeschreven giro-betaalkaarten en stortingen. We voeren de volgende veldinformatie in: NUMMER, BGNSTGDE, BEDRAG, ALGEBOEKT, STORTING. Om het bedrag te kunnen vaststellen dat op de rekening staat, voeren we de volgende regels in via het toetsenbord.

```
.SUM BEDRAG FOR STORTING TO STORT
17434.53
.SUM BEDRAG FOR .NOT.STORTING TO UITGAAF
15997.18
.STORE STORT-UITGAAF TO SALDO
1437.35
```

We hebben nu een nuttig resultaat berekent, namelijk het bedrag dat op de rekening staat, rechtstreeks vanaf het toetsenbord in drie vragen. We controleren nu de giro door de berekeningen van de computer te vergelijken met de afrekening via de volgende opdrachten:

```
.SUM BEDRAG FOR STORTING .AND.ALGEBOEKT TO GIROBIJ
16621.21
.SUM BEDRAG FOR .NOT.STORTING.AND.ALGEBOEKT TO GIROAF
15793.23
.STORE GIROBIJ-GIROAF to SALDO
827.98
```

Zo hebben we – met slechts drie instructies aan de computer – een tweede nuttig resultaat gekregen: een controle op de mening van de girodienst over ons saldo. De laatste uitkomst kan vergeleken worden met de afrekening.

In de laatste twee voorbeelden zouden we de gezochte sommen kunnen hebben verkregen zonder ze op te slaan. We zouden de uitkomst ook met potlood en papier of met een zakrekenmachine hebben kunnen berekenen. Dat zou kans op fouten gegeven hebben – het met de hand overnemen van de getallen. Het 'kladblokgeheugen' geeft ons de mogelijkheid de computer rechtstreeks te gebruiken bij het rekenen met gegevens uit de database. Indien de gegevens in de database juist zijn, is de kans op fouten vrijwel nul.

Mocht u het gevoel krijgen, dat dit allemaal gewoon te eenvoudig is om waar te zijn – dat u iets essentieels over het hoofd moet zien – maakt u zich dan niet bezorgd: het is echt gemakkelijk.

Veel van wat vandaag de dag over computers geschreven wordt, draagt aanzienlijk bij tot de opvatting dat computers moeilijk zijn. Leerboeken behandelen het onderwerp vanuit de denkwereld van iemand met een technische achtergrond. In een dergelijk leerboek zou de bespreking van een voorbeeldcommando met een .OR. aanzienlijk verschillen van onze behandeling. Het schermbeeld dat de computer geeft in antwoord op een dergelijk voorbeeldcommando zou in een leerboek beschreven kunnen worden als 'de relatie B:INVENTRS beperkt door het predikaat DRANK= 'SHERRY' .OR. DRANK='JENEVER". Een predikaat – voor het geval u het echt wilt weten – is 'een relatie tussen waarden behorend tot de domeinen'.

Het kan wel eens lijken alsof deskundigen gespecialiseerd 'jargon' gebruiken om hun positie en hun salaris te rechtvaardigen. Dat is toch niet helemaal waar. Het jargon stelt de specialisten in staat beter onderling informatie uit te wisselen.

Maar voeg de barrière die een dergelijk technisch taalgebruik opwerpt voor iemand die niet vertrouwd is met 'Het Computervak' bij de algemene opvatting dat computers vreemd, ontoegankelijk, moeilijk of iets dergelijks zijn, en u begrijpt waarom veel mensen denken: 'het kan gewoon niet zo gemakkelijk zijn'. In werkelijkheid is de moeilijkheid grotendeels gebrek aan vertrouwdheid. Computers zijn er niet alleen voor computerdeskundigen. Ze hebben onvoorstelbare mogelijkheden die gemakkelijk toegankelijk zijn voor iedereen die eraan behoefte heeft informatie op te slaan en te verbreiden.

De database slaat gegevens op. Het DBMS stelt u in staat informatie te onttrekken aan die gegevens. Als we een slijterij hebben, kunnen we gemakkelijk merken, dat we gevaarlijk weinig tequila in voorraad hebben en dat we verdrinken in de jenever. Als we de voorbeelddatabase met de inventaris samenvoegen met een database waarin de ontvangen artikelen staan vermeld, kunnen we bepalen, hoeveel we dit jaar verkocht hebben van elke soort drank en elke flesinhoud. In de loop van de tijd leren we dan de voorraad beter te beheren. Daaruit vloeit op zijn beurt een doelmatiger gebruik van geld voort en dus een beter rendement van onze investering. Database management systemen vinden natuurlijk vele andere toepassingen in het bedrijfsleven, in het onderwijs en bij de overheid. Waar het om gaat, is dit: het is gemakkelijk dit stuk gereedschap te ontwikkelen en het is nuttig. Het kan een reële hulp zijn, wat u ook doet. Het kan leuk zijn. Het kan u vrijmaken voor interessantere bezigheden. Het kan uw kijk op de zaken verbreden, waardoor u in staat zult zijn uw hele werkterrein vanuit één vast gezichtspunt te overzien.

Deel 3

DEEL DRIE

Deel Drie bespreekt allerlei soorten databases, waarbij we de aard van elk systeem onderzoeken en de onderlinge verschillen aangeven.
In Hoofdstuk 10 bespreken we computerlogica, wat op het eerste gezicht misschien een ingewikkeld en moeilijk onderwerp lijkt.

Technische en ingewikkelde terminologie zorgt er soms voor, dat computers geheimzinnig en/of moeilijk lijken. Die indruk is onjuist en misleidend – computers zijn in feite heel begrijpelijk en bovendien erg leuk. Deel Drie probeert enige leemten op te vullen en een paar vragen te beantwoorden, die gewone mensen over computers plegen te stellen. Daarmee zal hopelijk ons begrip van de computerwereld toenemen en zullen we in staat zijn verdere vooruitgang te boeken met onze computerdatabases.

8. DINGEN DIE U MISSCHIEN WILT WETEN

Een aantal vragen over database systemen voor microcomputers komt vaak terug. In dit en de volgende hoofdstukken gaan we proberen enkele van de meestgestelde vragen te beantwoorden. Wat is een machinetaalsysteem? Wat is een hiërarchische database? Wat is een netwerkdatabase? Wat is CODASYL? Wat zijn.....?

MACHINETAAL (of assembler) is de 'moedertaal' van de computer. De naam is eigenlijk nogal onnauwkeurig, het is de 'taal' die de computer intern gebruikt. Naast machinetaal zijn er talen 'van hoger niveau', zoals FORTRAN, COBOL, PASCAL en de APPLICATION DEVELOPMENT LANGUAGE (ADL) van dBASE II. Elk commando of elke instructie in deze talen is 'opgebouwd' uit vele instructies machinetaal.

Wanneer de computer een commando als DISPLAY uitvoert, voert hij in feite een groot aantal machinetaalinstructies uit. Talen van zeer hoog niveau als BASIC en ADL gebruiken gemiddeld 50 tot 100 machinetaalinstructies voor elke instructie in de taal van hoog niveau. Een database management systeem in machinetaal is 'opgebouwd' uit een groot aantal instructies in machinetaal.

Veel programmatuurpakketten die te koop zijn, zijn geschreven in een taal van hoog niveau, zoals PASCAL of BASIC. Dit wordt gedaan omdat het meestal gemakkelijker is het pakket te schrijven in een taal van hoog niveau en omdat het pakket dan 'overdraagbaar' wordt. Overdraagbaarheid houdt in, dat hetzelfde pakket gemakkelijk kan worden aangepast voor gebruik op verschillende soorten microcomputers.

Een programma in machinetaal is alleen bruikbaar op één bepaald soort microcomputer. Om het pakket op een ander soort computer te kunnen gebruiken, moet het speciaal herschreven worden.

Er wordt vaak aangenomen, dat een machinetaalprogramma de mogelijkheden van de machine doelmatiger gebruikt dan een programma in een taal van hoog niveau als BASIC. Dit is waarschijnlijk waar, als u een goed geschreven programma in machinetaal vergelijkt met een goed geschreven programma in een taal van hoog niveau.

Omdat het echter moeilijk is uit te maken of twee systemen even goed geschreven zijn, zou dit geen factor mogen zijn bij het kiezen van een van de systemen die op de markt zijn. Beide benaderingen van een programmatuursysteem hebben hun voordelen. Omdat u haast nooit een verkoper zijn systeem zult horen aanprijzen als 'middelmatig' of 'redelijk tot goed', zou u bij het maken van uw keuze waarschijnlijk als factor buiten beschouwing moeten laten, hoe goed een programma geschreven is. Wat echt belangrijk voor u is, is, of de programmatuur uw probleem kan oplossen.

SEQUENTIËLE EN RANDOM TOEGANG

Termen die u vaak tegenkomt in artikelen over databases en soms ook wel in advertenties, zijn SEQUENTIËLE TOEGANG en RANDOM TOEGANG. Deze termen hebben betrekking op de manier waarop de computer de gegevens ophaalt. Sequentiële toegang houdt in, dat de computer begint bij het eerste record en de hele databasefile in volgorde doorloopt, totdat hij het door u gewenste record vindt. Commando's uit vraagtalen, zoals DISPLAY en LOCATE, gebruiken sequentiële toegang.

.DISPLAY FOR NAME = 'PIETERS, ROB'

.LOCATE FOR NAME = 'PIETERS, ROB'

Wanneer in dit geval DISPLAY wordt gebruikt, zal de computer alle records in de database onderzoeken – te beginnen bij record 1 – in de volgorde van hun recordnummers (sequentiëel). Wanneer LOCATE wordt gebruikt, zal de computer de records in de database onderzoeken, te beginnen bij record 1, sequentiëel doorgaand totdat hij aankomt bij het record dat PIETERS, ROB bevat in het veld NAAM. Dit is sequentiële toegang tot een bepaald record. Merkt u op, dat de tijd die de computer nodig heeft om een bepaald record te vinden, afhangt van de plaats in de database, waar het record zich bevindt. Als de database groot is en het gewenste record zich aan het einde bevindt, zou het de computer verscheidene seconden kunnen kosten om het record te vinden. Als de computer de hele database kan 'lezen' in een minuut, is de gemiddelde toegangstijd 30 seconden.

De term RANDOM TOEGANG (willekeurige toegang) is een beetje misleidend. De term betekent niet, dat de computer op goed geluk de database doorbladert, totdat hij het gewenste record tegenkomt.

In feite betekent random toegang dat de computer rechtstreeks toegang heeft tot elk record in de database. Het woord random komt eigenlijk van het idee dat als u een willekeurig record uitkiest, de computer daar net zo snel kan komen als bij ieder ander record.

Een ruw voorbeeld van een database op papier, die is ontworpen voor random (rechtstreekse) toegang, is het telefoonboek. De namen in het telefoonboek staan op alfabetische volgorde. Wanneer u het telefoonnummer van een bepaald iemand zoekt, gebruikt u de alfabetische indeling van het boek om diens naam te vinden en zo het gewenste nummer. Als u via sequentiële toegang het nummer zou proberen te vinden, zou u beginnen bij de eerste naam in het telefoonboek en zou u het helemaal doorlopen, naam voor naam, totdat u de gewenste naam zou tegenkomen. Sequentiële toegang is eenvoudig, betrouwbaar maar betrekkelijk langzaam. Rechtstreekse (random) toegang kan een veel snellere manier opleveren om bij een bepaald record te komen.

PRIMAIRE EN SECUNDAIRE SLEUTELS

Als de computer random (rechtstreekse) toegang moet hebben tot een bepaald record, heeft hij een beetje hulp nodig. De gerichte 'hulp' die de computer meestal krijgt is een speciale 'PRIMAIRE SLEUTEL', die de computer heeft voor het record. De computer gebruikt deze SLEUTEL om rechtstreeks naar het record toe te gaan, waarbij hij hetzij bijhoudt, waar het record zich bevindt, hetzij dat kan berekenen. Wanneer vervolgens om het record gevraagd wordt, gaat de computer rechtstreeks naar de plaats van het record op de diskette en zet hij het in het hoofdgeheugen. De 'sleutelinformatie' wordt door de computer toegevoegd bij het maken van het record. Bij dBASE II is het RECORDNUMMER de 'primaire sleutel' die de computer toevoegt. Als u het recordnummer kent, kunt u rechtstreeks naar het record toegaan. Het opsporen van een record 'met de sleutel' gaat erg snel, maar de gebruiker moet wel eerst de 'SLEUTEL' kennen. Bij dBASE II betekent kennen van de 'sleutel' kennen van het RECORDNUMMER.

De primaire sleutel moet natuurlijk uniek zijn. Het is niet waarschijnlijk dat u zich de primaire sleutel kunt herinneren van alle records in een computerdatabase. Het is soms wel mogelijk bij een heel kleine database; als de database echter klein is, is rechtstreekse toegang niet nodig.

Een SECUNDAIRE SLEUTEL is de oplossing voor het probleem hoe u de primaire sleutel het beste kunt gebruiken. De database kan doorgaans een of meer secundaire sleutels hebben. 'Sleutels' die de computer zelf gebruikt, zijn vaak voor ons niet erg handig en 'sleutels' die voor ons belangrijk zijn, zijn meestal niet bruikbaar voor de computer die rechtstreekse toegang probeert te krijgen. Secundaire sleutels leggen de verbinding tussen computersleutels en onze eigen manier van denken.

Secundaire sleutels zorgen niet voor echt rechtstreekse toegang, maar ze komen daar wel bij in de buurt. De toegang komt veel sneller tot stand dan de gemiddelde toegangstijd bij sequentiële toegang. Tabellen zorgen voor de 'vertaling' van secundaire in primaire sleutels.

Een gewoon kookboek levert een goed voorbeeld van primaire en secundaire sleutels in het dagelijks leven.

De primaire sleutel is het bladzijdenummer. De secundaire sleutel(s) zijn de namen van de gerechten en de ingrediënten. De index is de tabel die de naam van het gerecht 'vertaalt' in een bladzijdenummer. Het bladzijdenummer leidt u naar het gewenste. Als u al wist dat het gewenste recept op bladzijde 123 stond, zou dat sneller gaan dan gebruik van de index. Gebruik van de index is echter een stuk sneller dan het kookboek doorbladeren. Misschien staat er meer dan één bladzijdenummer naast sommige trefwoorden in de index. Die trefwoorden zijn voorbeelden van niet unieke sleutels. Secundaire sleutels hoeven dan ook niet uniek te zijn.

Laten we veronderstellen dat we een database met leerlingen hebben. Elk record bevat de naam van de leerling, het adres, het telefoonnummer, het lokaal, de klas enzovoorts. We zouden graag toegang hebben tot de records van de leerlingen via de naam of de klas of het lokaalnummer. Om dit te doen, laten we deze drie velden optreden als secundaire sleutels. We hebben dan drie tabellen, voor elk van de drie sleutels een. Wanneer we het record van een bepaalde leerling willen hebben, maken we gebruik van de naamsleutel en vragen we de leerling bij zijn of haar naam op. De computer zoekt de naam in de tabel op, pakt het recordnummer en gebruikt dat om het record uit de database te halen.

Omdat de tabellen een bepaald doel dienen, kunnen ze zo opgezet worden dat ze een bijna rechtstreekse toegang geven tot het gewenste gegeven. Bij dBASE II wordt rechtstreekse toegang verschaft door indexfiles te gebruiken als tabellen. Het systeem zal de tabellen opzetten, wanneer u het commando INDEX gebruikt. De drie tabellen kunnen als volgt worden opgezet:

```
.USE B:SCHOOL
.INDEX ON NAAM TO B:NAAM
.INDEX ON LOKAAL TO B:LOKAAL
.INDEX ON KLAS TO B:KLAS
```

Deze commando's hebben drie tabellen opgebouwd, die u in de gelegenheid stellen de drie velden NAAM, LOKAAL en KLAS te gebruiken als secundaire sleutels. De drie tabellen hebben als filenamen B:NAAM, B:LOKAAL en B:KLAS. Het database systeem voegt .NDX toe aan het einde van de filenaam om de computer duidelijk te maken, dat dit een INDEXFILE is, die gebruikt moet worden om toegang te krijgen tot records via secundaire sleutels. Als u records moet opzoeken op de naam van de leerling, geeft u het commando

```
.USE B:SCHOOL INDEX B:NAAM
```

U hebt nu rechtstreekse toegang tot het record van iedere willekeurige leerling door enkel de naam van de leerling te geven met het commando FIND.

```
.FIND Aardvarken, Anton
```

Als Anton overigens de enige leerling op school zou zijn, wiens naam begon met Aa, dan hadden we kunnen gebruiken wat dezelfde uitkomst had gegeven. De computer heeft Antons record gevonden. Om dat record te zien te krijgen typt u gewoon het woord DISPLAY. Als u geïnteresseerd bent in de zesde klas, zou u zeggen

```
.USE B:SCHOOL INDEX B:KLAS
.FIND 6
```

Met deze twee commando's komt u terecht bij het eerste record van een leerling van de zesde klas. Alle leerlingen van de zesde klas zullen bij elkaar staan in de volgorde van hun recordnummers. Omdat ze bij elkaar staan, kunt u alle zesdeklassers laten weergeven met het commando

```
.DISPLAY WHILE KLAS = '6'
```

Wanneer u de database gebruikt samen met een indexfile (via secundaire sleutels) zal het de computer aanmerkelijk meer tijd kosten een commando als DISPLAY FOR KLAS='6' uit te voeren dan wanneer u de database gebruikt zonder indexfile. Omdat ze kunnen worden gebruikt om records te 'sorteren' in groepen, kunnen indexfiles worden gebruikt in plaats van commando's die de database echt een andere volgorde geven – zoals SORT. Behalve dat ze de feitelijke ordening van de database niet veranderen, werken indexcommando's veel sneller dan sorteercommando's.

U kunt ook een index maken op meer dan één veld tegelijk. Stel bijvoorbeeld dat u de leerlingen geordend wilt hebben op groep (lokaal en klas) en dat u wilt dat ze binnen een groep op alfabetische volgorde staan. U kunt de drie velden aan elkaar knopen als secundaire sleutel. Bijvoorbeeld:

```
.USE B:SCHOOL
.INDEX ON KLAS + LOKAAL + NAAM TO B:GROEP
```

Het plusteken verbindt de drie velden om ze één sleutel te laten vormen. Als u de hele database zou laten weergeven met het commando DISPLAY ALL, zouden de leerlingen verschijnen in volgorde van klas, lokaal bij lokaal voor elke klas en binnen elke dergelijke groep zouden ze alfabetisch staan. De computer bewaart deze files op diskette, zodat ze later opnieuw kunnen worden gebruikt, zonder dat u opnieuw een index hoeft te maken. Het opnieuw maken van een index kost wat tijd, vooral als de sleutel uit meerdere tekens bestaat en de database groot is. Het opnieuw maken van een index is alleen nodig als records worden toegevoegd of verwijderd of als in een record een van de sleutelvelden wordt veranderd.

Bij sommige database management systemen kan 'automatisch' voor opnieuw indexeren gezorgd worden. Bij dBASE II wordt dit gedaan, wanneer

u de indexfiles gebruikt, terwijl u records toevoegt, verandert of wist. In ons voorbeeld zal

.USE B:SCHOOL INDEX B:GROEP, B:NAAM, B:LOKAAL, B:KLAS

ervoor zorgen dat de indexfiles B:GROEP, B:NAAM, B:LOKAAL en B:KLAS worden bijgewerkt als een record wordt toegevoegd, één van de 'sleutelvelden' wordt veranderd of records worden verwijderd met de commando's DELETE en PACK.

Dit heeft ook nadelen. Meerdere indexfiles (meerdere secundaire sleutels) bijwerken kost veel tijd. Heel veel tijd. Niet alleen bij dBASE II is dat zo. Het bijwerken van meerdere tabellen met secundaire sleutels vergt bij alle database management systemen erg veel tijd. Daarom wordt meestal aanbevolen het gebruik van meerdere sleutels (meerdere indexfiles) zoveel mogelijk te vermijden. In verband hiermee is ook nog een nadeel dat een verandering die in de database wordt aangebracht zonder dat de verandering ook in de indexfile wordt opgenomen, de tabel ongeldig kan maken en daarmee alle computerbewerkingen die met gebruikmaking van die tabel zijn uitgevoerd.

Secundaire sleuteltabellen nemen ruimte in op diskette. Als u meer dan één diskettedrive hebt kunt u een indextabel in een andere diskettedrive zetten dan de database. De tabellen hoeven niet op dezelfde diskette te staan als de database. Ze moeten echter wel on-line zijn. Dit houdt in, dat de computer tegelijkertijd toegang moet hebben tot de database en tot de indexfile die u gebruikt. Een indexfile moet (net als een database) bij de meeste microcomputer database systemen in zijn geheel op één enkele diskette staan. U doet er goed aan daarmee rekening te houden bij het maken van een plan voor uw database systeem en ook wanneer u de computerapparatuur uitkiest, die uw database systeem moet ondersteunen.

FYSIEKE EN LOGISCHE RECORDS

In Hoofdstuk 1 hebben we de gang van zaken in een auto-onderdelenwinkel gebruikt om het idee van een database management systeem uit te leggen. In deze analogie kwamen de winkelbediende, zijn onderdelencatalogus en de magazijnstellingen met de auto-onderdelen overeen met het DBMS, terwijl de eigenlijke onderdelen analoog waren met de informatie die is opgeslagen in de database. Als u meer over database systemen begint te lezen, zult u de aanduidingen FYSIEKE RECORDS en LOGISCHE RECORDS tegenkomen. Fysieke records zijn de eigenlijke records met gegevens – die komen overeen met de auto-onderdelen uit de analogie. Logische records worden gevormd door de vermeldingen in de tabellen met secundaire sleutels (in de indexfiles), die de computer vertellen waar op diskette het fysieke record zich bevindt. Deze records komen overeen met de vermeldingen in de catalogi van de bediende, die hem vertellen waar de auto-onderdelen zich bevinden.

In Hoofdstuk 1 hebben we ook een bibliotheek gebruikt als analogie met een database management systeem. In het voorbeeld van de bibliotheek komen de fysieke records overeen met de boeken. De logische records komen overeen met de kaartjes in de catalogus.

In ons voorbeeld van de bibliotheek was er nog een andere verzameling logische records, die gebruikt werd om boeken te localiseren. Dat zijn de 'reserveringskaarten'. Het kan gebeuren dat een boek dat u uit de bibliotheek wilt hebben, is uitgeleend. De bibliothecaresse kan deze records raadplegen om u te vertellen, wanneer het boek teruggebracht zou moeten zijn. Als u dat wenst kan het boek voor u gereserveerd worden en krijgt u bericht, wanneer het boek is teruggebracht. Het zal dan een bepaalde tijd voor u worden vastgehouden.

RECORDS BLOKKEREN

Net als in het voorbeeld van de bibliotheek zou meer dan één gebruiker tegelijk een fysiek record nodig kunnen hebben. Meestal is het niet wenselijk dat meer dan één persoon tegelijk een database record gebruikt. Stel bijvoorbeeld eens dat de database de lege stoelen bevat op een bepaalde vlucht van een luchtvaartmaatschappij. Als twee agenten tegelijk die vluchtinformatie zouden gebruiken, zouden ze twee verschillende klanten de laatste stoel van de vlucht kunnen verkopen. Om te voorkomen dat iets dergelijks gebeurt, 'reserveert' het DBMS het record voor degene die het als eerste opvraagt. Wie het als tweede opvraagt, zal ervan in kennis gesteld worden, dat het record op het moment in gebruik is. Het DBMS zal het record reserveren voor gebruik door de tweede aanvrager, die het zal krijgen zodra het ter beschiking komt. Het record is zoals dat heet, 'geblokkeerd' voor iedereen die er toegang toe vraagt, totdat de eerste gebruiker het heeft vrijgegeven. De technische term hiervoor is BLOKKERING van het record.

Het is nu nog ongebruikelijk dat een database management systeem voor een microcomputer de mogelijkheid biedt records te blokkeren. De meeste huidige microcomputersystemen zijn ontworpen voor gebruik door slechts één persoon tegelijk – de personal computer. Er is echter een sterker wordende tendens microcomputers door meerdere mensen te laten delen. Dit maakt het mogelijk betrekkelijk dure en meestal onder zijn capaciteit werkende randapparatuur, zoals een printer, te laten 'delen'. Naarmate database management systemen meer en meer op microcomputersystemen gebruikt zullen gaan worden, zal ook dit het gebruik aanmoedigen van microcomputers door meerdere personen tegelijk. Dit zal zorgen voor een opzet waarbij het mogelijk is de informatie te 'delen'. Het snel en goedkoop kunnen delen van informatie is van grote waarde. Die mogelijkheid is de hoofdreden achter de ontwikkeling van database management systemen voor grote centrale computers. Het laat zich aanzien, dat die mogelijkheid zal leiden tot de ontwikkeling van apparatuur die gezamenlijk gebruik van informatie ondersteunt maar toch betaalbaar is voor de kleinere organisatie.

9. MEER DINGEN DIE U WILT WETEN

Database systemen zijn er in drie hoofdsoorten.

RELATIONELE

HIËRARCHISCHE

en

NETWERKSYSTEMEN

Het belangrijkste verschil tussen de drie is de manier waarop de gebruiker de organisatie van de gegevens ervaart. Wij denken over gegevens in termen van tabellen – met rijen en kolommen – in een relationeel systeem. Het schema van een hiërarchische organisatie is het beeld dat we voor ons zien bij het organisatie schema van een bedrijf. Wanneer we ons een netwerksysteem proberen voor te stellen, denken we aan het gemeenschappelijke organisatieschema van twee bedrijven die net een fusie zijn aangegaan. De meeste database management systemen zijn of relationeel, of hebben de vorm van een bepaalde netwerkversie die CODASYL heet.

DE RELATIONELE DATABASE

Dit boek richt zich vooral op de RELATIONELE database, omdat die zo gemakkelijk aansluit bij de ervaring van alledag. De meeste database management systemen voor microcomputers zijn variaties op de RELATIONELE grondgedachte. Het relationele database systeem is precies wat het lijkt te zijn. De gegevens worden behandeld en opgeslagen op wat voor de meeste mensen een natuurlijke manier is. Bij gebruik van een relationeel systeem kan iemand die geen ervaring heeft en die niet vertrouwd is met computersystemen, toch nuttige dingen doen met niet al te veel moeite.

Een voorbeeld van een eenvoudige relationele database (gehaald uit onze inventarislijst van de slijterij in Hoofdstuk 2) staat in Figuur 9-1.

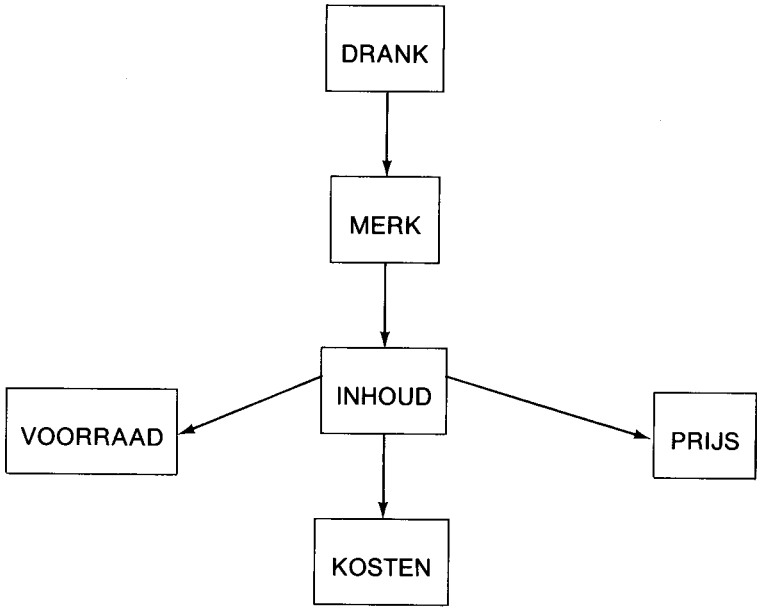
Drank	Merk	Inhoud	Vrd	Kosten	Prijs
SHERRY	ANDALUCIA	1.0 L	23	5.59	9.31
SHERRY	ANDALUCIA	2.0 L	7	9.78	16.30
SHERRY	ANDALUCIA	0.5 L	88	2.74	4.56
WODKA	RUSSKI	1.0 L	35	3.78	6.30
WODKA	RUSSKI	2.0 L	9	7.95	13.25
WODKA	RUSSKI	0.5 L	75	1.49	2.48
WHISKEY	SOUTHERN RYE	1.0 L	32	5.11	8.51
WHISKEY	OLD IRISH	0.5 L	44	1.98	3.30
WHISKEY	OLD IRISH	1.0 L	19	5.29	8.81
WHISKEY	THE NEW SOUTH	1.0 L	4	7.49	12.48
JENEVER	SCHIEDAM	0.5 L	5	0.99	1.65
JENEVER	SCHIEDAM	0.8 L	22	1.78	2.96
JENEVER	SCHIEDAM	1.0 L	21	3.50	5.83
JENEVER	SCHIEDAM	2.5 L	3	6.89	11.48
JENEVER	SCHIEDAM	2.0 L	5	6.47	10.78

Figuur 9-1. Voorbeeld van een relationele database

Deze relationele database ziet er net zo uit als wanneer u een inventarislijst zou maken met potlood en papier. Elk record heeft een vaste lengte. Elk veld binnen het record heeft altijd dezelfde breedte.

DE HIËRARCHISCHE DATABASE

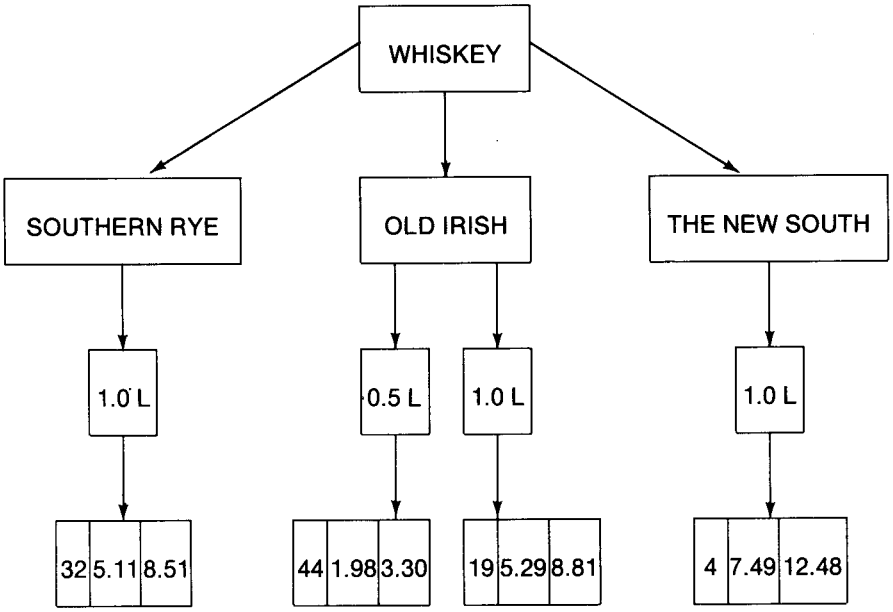
Bij HIËRARCHISCHE database systemen moet u de gegevens gerangschikt denken als in een hiërarchie geordend. Een diagram van een hiërarchische database lijkt op een organisatieschema. Het lijkt ook op een boom op zijn kop. Hiërarchische structuren worden vaak boomstructuren genoemd. De database van de slijterij zou weergegeven kunnen worden als afgebeeld in Figuur 9-2.



Figuur 9-2. Hiërarchische voorstelling van inventaris slijterij

Wanneer gegevens op deze manier worden weergegeven, komt de hiërarchische relatie tevoorschijn. Elk gegeven is ondergeschikt aan een ander gegeven (behalve natuurlijk het gegeven in het bovenste vak). In plaats van dat het 'record' een eenvoudige verzameling velden is, is het een verzameling subrecords of 'segmenten'. Elk vak in het voorbeeld is een 'segment' of een stuk van het record. Een segment kan meer dan één veld bevatten. De onderste drie vakken zouden bijvoorbeeld samengebracht kunnen worden in één enkel segment.

Het gebruik van het woord record is een beetje ongelukkig, omdat het moeilijk is bij onze manier van denken het ene soort 'record' van het andere te scheiden. In deze versie van onze slijterijdatabase hebben we vier 'records': één voor elk van de vier verschillende soorten drank, die in de relationele database B:INVENTRS zijn opgenomen: sherry, wodka, whiskey en jenever. Het hiërarchische 'whiskey' record, dat overeenkomt met de 'whiskey' regels in onze relationele database, is afgebeeld in Figuur 9-3.



Figuur 9-3. Een hiërarchisch whiskeyrecord

Bij een dergelijk systeem hangt elk segment af van een ander segment, zodat geen segment mag afhangen van meer dan één ander segment. Een segment mag echter wel meer dan één segment onder zich hebben. Segmenten met één of meer segmenten onder zich worden vaak ouders genoemd. In het voorbeeld van Figuur 9-3 is whiskey de ouder van Southern Rye, Old Wyoming en The New South. Deze zijn alle op hun beurt ouders (of bezitters) van de verschillende INHOUDssegmenten. Ze zijn ook de KINDEREN van 'whiskey'.

Laten we nu eens kijken, hoe dit werkt. Elk segment heeft een eigen herkenningscode bij zich. Ieder segment heeft een andere code (de code is een primaire sleutel). De code geeft het soort segment aan en het rangnummer (net zoiets als het recordnummer bij dBASE II).

De vier segmenten DRANK bevatten elk verscheidene wijzers. Elke wijzer stuurt de computer naar een segment MERK, dat 'hoort' bij dat segment DRANK. Elk van deze segmenten MERK bevat op zijn beurt wijzers die de computer sturen naar de verschillende segmenten INHOUD, die bij dat MERK horen.

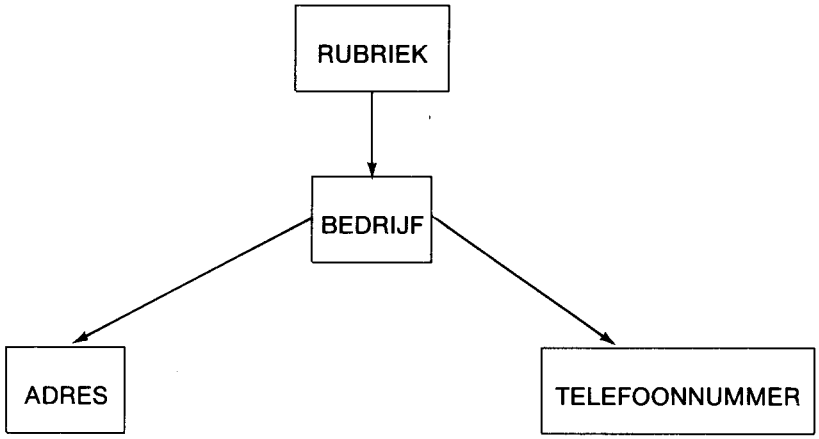
Op hun beurt bevatten die elk wijzers die de computer naar de segmenten met VOORRAAD, KOSTEN en PRIJS sturen. Deze segmenten bevatten geen wijzers. Wijzers stellen de computer in staat rechtstreeks van het ene naar het andere segment af te dalen om het record samen te stellen.

Oppervlakkig bezien lijkt dit een moeilijke manier om iets gemakkelijks gedaan te krijgen. In een relationele database hebben we deze wijzers niet nodig om alle records samen te stellen. In ons relationele voorbeeld gebruiken we echter vier records voor de 'whiskey-inventaris'. In de hiërarchische versie hebben we maar één 'whiskeysegment' nodig plus een paar wijzers. Als de inventarislijst van onze slijterij uit duizend artikelen bestaat en er maar tien soorten drank zijn, dan sparen we met het hiërarchische systeem een heleboel dubbel werk uit.

Het uitsparen van een heleboel dubbel werk lijkt een goede zaak. Maar toch wringt de schoen ergens. Ten eerste wordt het dubbele werk uitgespaard ten koste van de eenvoud. Veel toepassingen zullen gemakkelijk in dit raamwerk passen, vele andere echter niet. Bovendien moet u van tevoren beslissen, wat uw toepassingen zullen zijn.

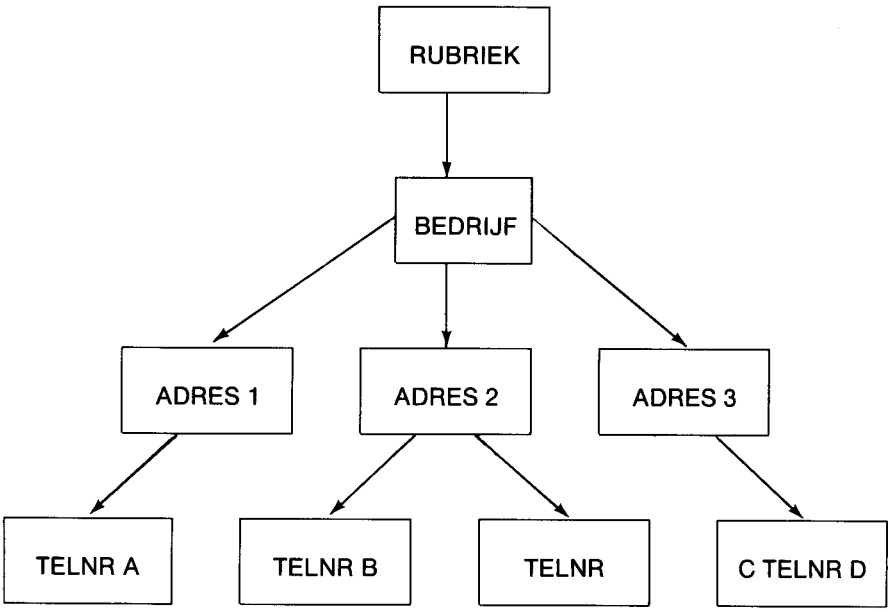
Het feit dat een segment maar bij één ouder kan horen, is één nogal duidelijk nadeel van het hiërarchische systeem. Stel eens dat we een personeelsdatabase hebben, die hiërarchisch is opgezet. Twee van onze werknemers trouwen en krijgen een kind. Het segmentrecord voor dat kind kan in de database niet bij de werknemersrecords van beide ouders horen. Dat is natuurlijk belachelijk en er zijn vele lapmiddelen ontwikkeld om dit voorbeeld aan te kunnen. Niettemin is het tekenend voor een klassiek gebrek van het hiërarchische systeem.

De Gouden Gids is een ruwe analogie voor een hiërarchische database uit de wereld van de databases op papier. Als we de Gouden Gids weergeven als afgebeeld in Figuur 9-4, wordt de hiërarchische aard duidelijk.



Figuur 9-4. Hiërarchische voorstelling van de Gouden Gids

ADRES en TELEFOONNUMMER horen bij het bedrijf, het BEDRIJF hoort op zijn beurt bij de RUBRIEK. Andersom is RUBRIEK de bezitter van BEDRIJF, BEDRIJF op zijn beurt de bezitter van zowel ADRES als TELEFOONNUMMER. Een segment kan niet bij meer dan één bezitter horen. Een bezitter kan echter vele andere segmenten bezitten.



Figuur 9-5

In tegenstelling tot bij de relationele database kunnen records een verschillende afmeting hebben. Sommige records zouden bijvoorbeeld meerdere telefoonnummers kunnen hebben, terwijl andere er maar één hebben. Een relationele database kan dit ook wel aan, maar dan moet u of opslagruimte 'verknoeien' of slim zijn. Voor bedrijven die meer dan één filiaal en/of telefoonnummer hebben, zou de structuur geschematiseerd kunnen worden als afgebeeld in Figuur 9-5.

Stel om hiervan nog eens een betere indruk te krijgen, dat we een record hebben met autodealers. Het segment RUBRIEK bevat dan het opschrift 'Autodealers' plus wijzers die de computer naar alle segmenten sturen, die de namen bevatten van autodealers. Een van die dealers is 'Autobedrijf Vroem Vroem'. Het segment van 'Autobedrijf Vroem Vroem' bevat de naam van het bedrijf en een verzameling wijzers die de computer naar de vestigingen van het bedrijf sturen. Het database systeem kan de segmentkop gebruiken om te zorgen voor een ordening binnen een record (om bijvoorbeeld de bedrijfsnamen in alfabetische volgorde te houden). Het segment van elk adres bevat het adres en de wijzers naar de telefoonnummers. Een segment bevat altijd wijzers die de computer naar de lagere segmenten leiden. SOMS bevat een segment een wijzer die de computer naar het bezittende segment leidt.

Er zijn een paar mogelijke tekortkomingen bij een hiërarchisch database systeem – bezien vanuit de gebruiker. In de eerste plaats: als we een alfabetisch overzicht zouden willen hebben van alle bedrijven in de database, dan zou dat naar alle waarschijnlijkheid heel wat moeite kosten. In de tweede plaats is het zeer wel mogelijk, dat Autobedrijf Vroem Vroem tot meer dan één rubriek behoort. Autobedrijf Vroem Vroem kan best beschikken over een reparatiewerkplaats, een uitdeukinrichting, een onderdelenhandel en kan behalve nieuwe auto's ook best tweede-hands auto's verkopen. In een hiërarchisch systeem mag een kind maar één ouder hebben. Het autobedrijf uit ons voorbeeld moet in vijf rubrieken vermeld worden. Om dit voor elkaar te krijgen, moet het vijfmaal worden ingevoerd, voor iedere rubriek eenmaal.

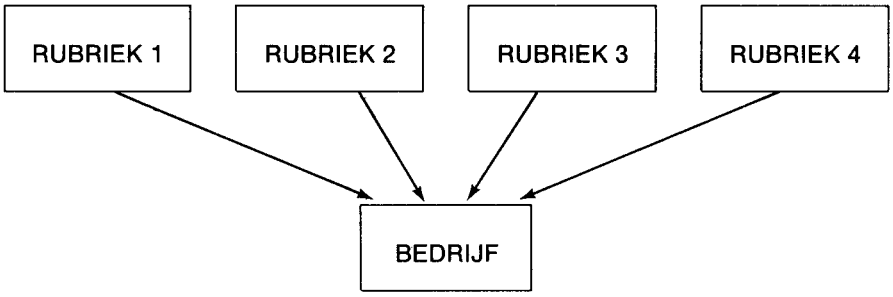
DE NETWERKDATABASE

NETWERK database management systemen lijken enigszins op hiërarchische systemen, met als belangrijk verschil, dat onder zekere voorwaarden een 'kind' meer dan één 'ouder' kan hebben. Een ander verschil is, dat een 'ouder-kind verhouding' (bijvoorbeeld RUBRIEK-BEDRIJF) kan worden omgekeerd. Tenslotte is de terminologie anders dan bij de hiërarchische en de relationele systemen.

De naam CODASYL wordt vaak gebruikt als synoniem voor het begrip NETWERK. Dit doet men, omdat de meest toegepaste NETWERK database systemen uitgaan van een in Amerika voorgestelde standaard voor databases, welke is ontwikkeld door de Data Base Task Group (DBTG) van de COnference on DAta SYstems Languages. CODASYL is de organisatie die de computertaal COBOL heeft ontwikkeld. De netwerkdatabase gaat uit van het begrip verzameling (uit de moderne wiskunde). De netwerkdatabase is nog ingewikkelder dan de hiërarchische, maar biedt ook meer mogelijkheden.

Bij het NETWERK database systeem is de database opgebouwd uit een stel verzamelingen. Elke verzameling bestaat uit records. Een record is hetzelfde als bij een relationeel systeem, behalve dat de lengte niet steeds hetzelfde hoeft te zijn. Een record kan bij meerdere verzamelingen horen. Een verzameling is een groep dingen van hetzelfde soort. Er zijn verschillende bedrijven in elke rubriek. Daarom horen alle bedrijven in een rubriek tot de verzameling van die rubriek. In de Gouden Gids behoren de autodealers zoals Autobedrijf Vroem Vroem tot de verzameling autodealers. Autobedrijf Vroem Vroem is een zogenaamd element van de verzameling autodealers. Het bezittende record is Autodealers.

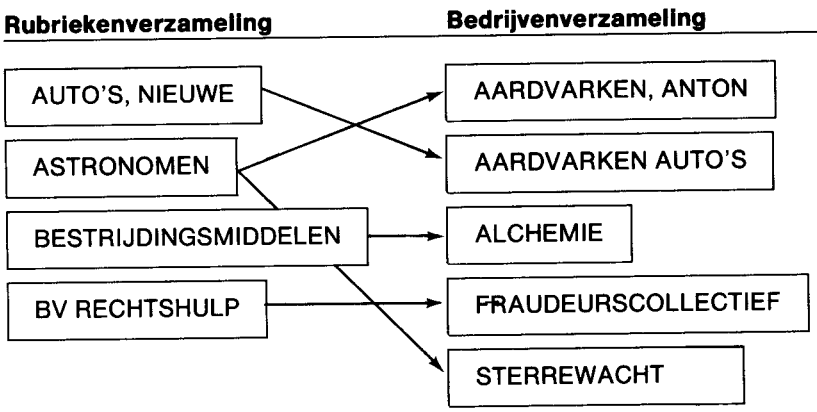
Elke verzameling moet een bezitterrecord hebben. Een verzameling mag uit slechts één record bestaan. Een record kan niet horen bij twee bezitters van hetzelfde verzamelingstype. De onderstaande situatie is dus niet toegestaan.



Figuur 9-6

Dit is natuurlijk vervelend, want we weten al, dat een bepaald bedrijf in de Gouden Gids onder een heleboel rubrieken kan voorkomen. Het probleem daarbij is, dat we een toestand hebben, waarin het mogelijke aantal betrekkingen zeer groot kan zijn. Het kan zo groot worden, dat ook de grootste computer het moet afleggen. Nu is de netwerkdatabase in zijn element als het gaat om betrekkingen van één ding tot vele. Het gaat er dus om de verzamelingen een zodanige opbouw te geven, dat alle betrekkingen die vorm hebben.

Een in het oog springende eigenschap van het hiërarchische database systeem was, dat een alfabetische lijst van alle bedrijven moeilijk te maken zou kunnen zijn, wanneer BEDRIJF afhankelijk was van RUBRIEK. Dit kwam doordat de BEDRIJVEN per rubriek op alfabetische volgorde staan en alleen via de rubriek toegankelijk zijn. Bij een netwerksysteem kan BEDRIJF van RUBRIEK afhangen en tegelijkertijd RUBRIEK van BEDRIJF.



Figuur 9-7

Bij netwerkdatabases wordt een heel andere terminologie gebruikt dan bij de hiërarchische of bij de relationele database systemen. Een 'gegeven' lijkt op wat wij een veld genoemd hebben. In het voorbeeld van de hiërarchische database zouden meerdere telefoonnummers bij één adres kunnen horen. Dit heet hier een 'vectorieel gegevensaggregaat'. Er zouden ook meerdere adressen bij één en hetzelfde bedrijf kunnen horen, waarbij elk adres minstens één telefoonnummer heeft. Een adres plus telefoonnummer heet een groep. Als er meer dan één groep is, heet dat een weerkerende groep.

HIËRARCHISCHE en NETWERK database systemen verschillen in opbouw nogal van het RELATIONELE systeem. De manier waarop gegevensrecords bij deze systemen zijn opgebouwd, is veel ingewikkelder dan bij het relationele systeem. Deze eerste twee systemen zijn goed geschikt voor grote, ingewikkelde database systemen. Ze worden daarom op grote schaal gebruikt op grote computers. Dit is zo omdat ze erg efficiënt kunnen zijn bij het gebruik van de middelen die de computer biedt – CPU tijd en hoofdgeheugen. De doelmatigheid die ze mogelijk bieden, kan buitengewoon belangrijk worden, wanneer de database tienduizenden of honderdduizenden records bevat.

Omdat het functioneren van een grote centrale computer al gauw duizenden guldens per uur kan kosten, is het duidelijk, dat doelmatigheid hier zijn waarde heeft. De kosten van de beroepsprogrammeurs die met deze database systemen werken, worden gemakkelijk goedgemaakt door de door hen teweeggebrachte vermindering op de kosten van het gebruik van dit soort computer. Het is niet zo duidelijk of dit soort programmaproductiviteit ook zo waardevol is bij databases die op microcomputers draaien.

Zoals al eerder gezegd, kan elk van de drie database systemen alle functies vervullen, die van een database gevraagd worden. Elk heeft zijn sterke en zwakke zijden: de hiërarchische en netwerksystemen bieden de gebruiker efficiency en snelheid. Ze zijn zuinig bij het gebruik van de mogelijkheden van de computer. Ze zijn echter ingewikkeld en nogal star. Ze zijn ontwikkeld voor gebruik op grote centrale computers, waarbij miljarden bytes on-line schijfopslag niet ongewoon zijn. Een miljard bytes komt overeen met duizend tot vierduizend diskettes van 8 inch. Alleen al het lezen daarvan met de snelheid van diskette-lezen zou een half miljoen seconden duren. Dat zijn DAGEN. Als u een echt grote database nodig hebt, dan is een database systeem op een microcomputer misschien wel niets voor u. Misschien hebt u werkelijk hetzij een hiërarchisch, hetzij een netwerk database systeem nodig en een grote computer. Hier komt de planning weer om de hoek kijken.

Elk van de drie soorten database systemen heeft toepassingen waarvoor het het meest geschikt is. Elk heeft zijn sterke en zijn zwakke zijden, ook al kunnen ze alle drie alle taken van een database vervullen. De netwerk- en de hiërarchische benadering vereisen dat u de gegevens als netwerk, respectievelijk als hiërarchie inbrengt. Een gevolg is, dat u uw database zo moet 'opzetten' dat de gegevens ook op de gewenste manier gebruikt kunnen worden. Voor u houdt dit in, dat u bij het 'maken' van de database al behoort te weten, hoe u hem zult gebruiken. Bij de relationele benadering echter, waar moet u de gegevens de vorm moet geven van tabellen met rijen en kolommen, kan de gebruikswijze van de gegevens later bepaald worden; in de praktijk een belangrijk verschil.

10. EEN STUKJE LOGICA

Computers en database management systemen zijn gestoeld op het gebruik van logica. De meeste database systemen voor microcomputers zijn zo ontworpen, dat dit gebruik van logica op een natuurlijke manier gebeurt. Het gebruik van logica is niet moeilijk – het is zelfs wel leuk. Hoe u er ook over moge denken, het begrijpen van de logica van de computer stelt u in staat veel meer profijt te hebben van uw computer en van uw database management systeem.

Er zijn drie veelgebruikte logische termen, die worden aangeduid als logische operatoren.

.AND.
.OR.
.NOT.

Zoals al eerder in dit boek gezegd, lijken deze woorden erg op hun evenbeelden in gewoon Engels. De punten aan weerszijden van het woord behoren tot de logische operator.

Om het gebruik van deze termen toe te lichten gebruiken we de inventarislijst van de slijterij uit het voorbeeld in Hoofdstuk 2. Die database staat in Figuur 10-1.

Drank	Merk	Inhoud	Vrd	Kosten	Prijs
SHERRY	ANDALUCIA	1.0 L	23	5.59	9.31
SHERRY	ANDALUCIA	2.0 L	7	9.78	16.30
SHERRY	ANDALUCIA	0.5 L	88	2.74	4.56
WODKA	RUSSKI	1.0 L	35	3.78	6.30
WODKA	RUSSKI	2.0 L	9	7.95	13.25
WODKA	RUSSKI	0.5 L	75	1.49	2.48
WHISKEY	SOUTHERN RYE	1.0 L	32	5.11	8.51
WHISKEY	OLD IRISH	0.5 L	44	1.98	3.30
WHISKEY	OLD IRISH	1.0 L	19	5.29	8.81
WHISKEY	THE NEW SOUTH	1.0 L	4	7.49	12.48
JENEVER	SCHIEDAM	0.5 L	5	0.99	1.65
JENEVER	SCHIEDAM	0.8 L	22	1.78	2.96
JENEVER	SCHIEDAM	1.0 L	21	3.50	5.83
JENEVER	SCHIEDAM	2.5 L	3	6.89	11.48
JENEVER	SCHIEDAM	2.0 L	5	6.47	10.78

Figuur 10-1. Inventarislijst slijterij

Stel dat u alle gegevens wilt zien voor WHISKEY en JENEVER. Uw eerste impuls is dan het vraagcommando te schrijven als:

.DISPLAY FOR DRANK = 'JENEVER'.AND.DRANK = 'WHISKEY'

Dit werkt helaas niet. Dit commando geeft de computer opdracht alle records te tonen met in het veld DRANK *zowel* WHISKEY *als* JENEVER. Die zijn er niet. Computerlogica heeft betrekking op één record tegelijk – niet op de hele database. Het juiste commando is:

.DISPLAY FOR DRANK = 'JENEVER'.OR.DRANK = 'WHISKEY'

Het gedeelte van de voorbeelddatabase, waarop dit betrekking heeft, is te zien in Figuur 10-2 als het niet-aangestreepte gebied.

Drank	Merk	Inhoud	Vrd	Kosten	Prijs
SHERRY	ANDALUCIA	1.0 L	23	5.59	9.31
SHERRY	ANDALUCIA	2.0 L	7	9.78	16.30
SHERRY	ANDALUCIA	0.5 L	88	2.74	4.56
WODKA	RUSSKI	1.0 L	35	3.78	6.30
WODKA	RUSSKI	2.0 L	9	7.95	13.25
WODKA	RUSSKI	0.5 L	75	1.49	2.48
WHISKEY	SOUTHERN RYE	1.0 L	32	5.11	8.51
WHISKEY	OLD IRISH	0.5 L	44	1.98	3.30
WHISKEY	OLD IRISH	1.0 L	19	5.29	8.81
WHISKEY	THE NEW SOUTH	1.0 L	4	7.49	12.48
JENEVER	SCHIEDAM	0.5 L	5	0.99	1.65
JENEVER	SCHIEDAM	0.8 L	22	1.78	2.96
JENEVER	SCHIEDAM	1.0 L	21	3.50	5.83
JENEVER	SCHIEDAM	2.5 L	3	6.89	11.48
JENEVER	SCHIEDAM	2.0 L	5	6.47	10.78

Figuur 10-2

Bij een goed gebruik van de .AND. operator wordt het gemeenschappelijke deel opgespoord van twee groepen records. Om bijvoorbeeld te bepalen, welke records halve-literflessen whiskey bevatten, is het commando:

.DISPLAY FOR DRANK = 'WHISKEY'.AND.INHOUD = '0.5 L'

De volgorde van de records doet er niet toe. Dezelfde uitkomst wordt verkregen met:

.DISPLAY FOR INHOUD = '0.5 L'.AND.DRANK = 'WHISKEY'

Als we .OR. hadden gebruikt in plaats van .AND. in het laatste voorbeeld, zouden we een heel andere uitkomst hebben gekregen.

.DISPLAY FOR INHOUD = '0.5 L'.OR.DRANK = 'WHISKEY'

Bij deze opbouw van het commando krijgt u het niet-aangestreepte gebied in Figuur 10-3.

Drank	Merk	Inhoud	Vrd	Kosten	Prijs
SHERRY	ANDALUCIA	1.0 L	23	5.59	9.31
SHERRY	ANDALUCIA	2.0 L	7	9.78	16.30
SHERRY	ANDALUCIA	0.5 L	88	2.74	4.56
WODKA	RUSSKI	1.0 L	35	3.78	6.30
WODKA	RUSSKI	2.0 L	9	7.95	13.25
WODKA	RUSSKI	0.5 L	75	1.49	2.48
WHISKEY	SOUTHERN RYE	1.0 L	32	5.11	8.51
WHISKEY	OLD IRISH	0.5 L	44	1.98	3.30
WHISKEY	OLD IRISH	1.0 L	19	5.29	8.81
WHISKEY	THE NEW SOUTH	1.0 L	4	7.49	12.48
JENEVER	SCHIEDAM	0.5 L	5	0.99	1.65
JENEVER	SCHIEDAM	0.8 L	22	1.78	2.96
JENEVER	SCHIEDAM	1.0 L	21	3.50	5.83
JENEVER	SCHIEDAM	2.5 L	3	6.89	11.48
JENEVER	SCHIEDAM	2.0 L	5	6.47	10.78

Figuur 10-3

Stel dat we die records willen opvragen, die halve-literflessen zijn met of whiskey of jenever. De manier om dit commando te geven is:

.DISPLAY FOR (DRANK = 'WHISKEY'.OR.DRANK = 'JENEVER').AND.INHOUD = '0.5 L'

De records die aan deze criteria voldoen, zijn te zien in de niet-aangestreepte gebieden van Figuur 10-4.

Drank	Merk	Inhoud	Vrd	Kosten	Prijs
SHERRY	ANDALUCIA	1.0 L	23	5.59	9.31
SHERRY	ANDALUCIA	2.0 L	7	9.78	16.30
SHERRY	ANDALUCIA	0.5 L	88	2.74	4.56
WODKA	RUSSKI	1.0 L	35	3.78	6.30
WODKA	RUSSKI	2.0 L	9	7.95	13.25
WODKA	RUSSKI	0.5 L	75	1.49	2.48
WHISKEY	SOUTHERN RYE	1.0 L	32	5.11	8.51
WHISKEY	OLD IRISH	0.5 L	44	1.98	3.30
WHISKEY	OLD IRISH	1.0 L	19	5.29	8.81
WHISKEY	THE NEW SOUTH	1.0 L	4	7.49	12.48
JENEVER	SCHIEDAM	0.5 L	5	0.99	1.65
JENEVER	SCHIEDAM	0.8 L	22	1.78	2.96
JENEVER	SCHIEDAM	1.0 L	21	3.50	5.83
JENEVER	SCHIEDAM	2.5 L	3	6.89	11.48
JENEVER	SCHIEDAM	2.0 L	5	6.47	10.78

Figuur 10-4

Stel dat u per ongeluk de haakjes in het laatste voorbeeld vergeten had. Het commando zou er zo uit hebben gezien:

```
.DISPLAY FOR DRANK = 'WHISKEY'.OR.DRANK = 'JENEVER'.AND.IN-
HOUD = '0.5 L'
```

Het schermbeeld dat hiervan de – volledig andere – uitkomst is, is afgebeeld in Figuur 10-5. Dit is zo, omdat de .AND. operator voorrang heeft op de .OR. operator.

Drank	Merk	Inhoud	Vrd	Kosten	Prijs
SHERRY	ANDALUCIA	1.0 L	23	5.59	9.31
SHERRY	ANDALUCIA	2.0 L	7	9.78	16.30
SHERRY	ANDALUCIA	0.5 L	88	2.74	4.56
WODKA	RUSSKI	1.0 L	35	3.78	6.30
WODKA	RUSSKI	2.0 L	9	7.95	13.25
WODKA	RUSSKI	0.5 L	75	1.49	2.48
WHISKEY	SOUTHERN RYE	1.0 L	32	5.11	8.51
WHISKEY	OLD IRISH	0.5 L	44	1.98	3.30
WHISKEY	OLD IRISH	1.0 L	19	5.29	8.81
WHISKEY	THE NEW SOUTH	1.0 L	4	7.49	12.48
JENEVER	SCHIEDAM	0.5 L	5	0.99	1.65
JENEVER	SCHIEDAM	0.8 L	22	1.78	2.96
JENEVER	SCHIEDAM	1.0 L	21	3.50	5.83
JENEVER	SCHIEDAM	2.5 L	3	6.89	11.48
JENEVER	SCHIEDAM	2.0 L	5	6.47	10.78

Figuur 10-5

De logische bewerking die de niet-aangestreepte gebieden in Figuur 10-4 tot gevolg heeft, wordt beschreven door het commando:

```
.DISPLAY FOR (DRANK = 'WHISKEY'.OR.DRANK = 'JENEVER').AND.IN-
HOUD = '0.5 L'
```

Stel nu, dat we in feite al het andere hadden willen hebben. Dan kan het rommelig worden als u de logica probeert uit te schrijven. U kunt dit echter op een gemakkelijke manier voor elkaar krijgen door de .NOT. operator te gebruiken. Het commando wordt

```
.DISPLAY FOR .NOT.((DRANK = 'WHISKEY'.OR.DRANK = 'JENEVE-
R').AND .INHOUD = '0.5 L')
```

De hele logische uitdrukking wordt tussen haakjes gezet om de computer duidelijk te maken, dat de .NOT. betrekking heeft op alles. De uitkomst is te zien in de niet-aangestreepte gebieden van Figuur 10-6.

Drank	Merk	Inhoud	Vrd	Kosten	Prijs
SHERRY	ANDALUCIA	1.0 L	23	5.59	9.31
SHERRY	ANDALUCIA	2.0 L	7	9.78	16.30
SHERRY	ANDALUCIA	0.5 L	88	2.74	4.56
WODKA	RUSSKI	1.0 L	35	3.78	6.30
WODKA	RUSSKI	2.0 L	9	7.95	13.25
WODKA	RUSSKI	0.5 L	75	1.49	2.48
WHISKEY	SOUTHERN RYE	1.0 L	32	5.11	8.51
WHISKEY	OLD IRISH	0.5 L	44	1.98	3.30
WHISKEY	OLD IRISH	1.0 L	19	5.29	8.81
WHISKEY	THE NEW SOUTH	1.0 L	4	7.49	12.48
JENEVER	SCHIEDAM	0.5 L	5	0.99	1.65
JENEVER	SCHIEDAM	0.8 L	22	1.78	2.96
JENEVER	SCHIEDAM	1.0 L	21	3.50	5.83
JENEVER	SCHIEDAM	2.5 L	3	6.89	11.48
JENEVER	SCHIEDAM	2.0 L	5	6.47	10.78

Figuur 10-6

Laten we nu aannemen, dat we alle records met whiskey en met jenever willen zien voor alle inhouden, behalve literflessen. Dit gaat met

```
.DISPLAY FOR (DRANK = 'WHISKEY'.OR.DRANK = 'JENEVER).AND.-
.NOT .INHOUD = '1.0 L'
```

De records die hierbij betrokken zijn, worden getoond in het niet-aangestreepte deel van Figuur 10-7.

Drank	Merk	Inhoud	Vrd	Kosten	Prijs
SHERRY	ANDALUCIA	1.0 L	23	5.59	9.31
SHERRY	ANDALUCIA	2.0 L	7	9.78	16.30
SHERRY	ANDALUCIA	0.5 L	88	2.74	4.56
WODKA	RUSSKI	1.0 L	35	3.78	6.30
WODKA	RUSSKI	2.0 L	9	7.95	13.25
WODKA	RUSSKI	0.5 L	75	1.49	2.48
WHISKEY	SOUTHERN RYE	1.0 L	32	5.11	8.51
WHISKEY	OLD IRISH	0.5 L	44	1.98	3.30
WHISKEY	OLD IRISH	1.0 L	19	5.29	8.81
WHISKEY	THE NEW SOUTH	1.0 L	4	7.49	12.48
JENEVER	SCHIEDAM	0.5 L	5	0.99	1.65
JENEVER	SCHIEDAM	0.8 L	22	1.78	2.96
JENEVER	SCHIEDAM	1.0 L	21	3.50	5.83
JENEVER	SCHIEDAM	2.5 L	3	6.89	11.48
JENEVER	SCHIEDAM	2.0 L	5	6.47	10.78

Figuur 10-7

U kunt vaak eigenschappen van de vraagtaal gebruiken om bepaalde nogal ingewikkelde logica te vermijden.

```
.DISPLAY FOR DRANK$'JENEVER   , WHISKEY   '
```

zal bijvoorbeeld dezelfde uitkomst leveren als

```
.DISPLAY FOR DRANK='JENEVER'.OR.DRANK='WHISKEY'
```

De eerste opdracht vertelt de computer alle records weer te geven, waarvan de inhoud van het veld DRANK vervat is in de 'tekenketen'. De spaties staan er, omdat de computer hier het hele veld DRANK zal vergelijken met de tekenketen. Het veld heeft tien tekens. Als u de spaties had weggelaten, zou de computer geen overeenkomst hebben gevonden met de inhoud van enig veld. Spaties hebben voor de computer evenveel betekenis als alle andere tekens.

Als u alles wilt zien behalve JENEVER of WHISKEY, dan is het juiste commando

```
.DISPLAY FOR .NOT.DRANK=$'JENEVER   , WHISKEY   '
```

Het dollarteken is een verkorte manier om te zeggen 'vervat in'. Dat dollarteken wordt soms aangeduid als tekenketenoperator. De extra spaties staan er, omdat het veld DRANK 10 posities heeft (breedte=10). Als de extra spaties niet geplaatst werden, zou de computer niet in staat zijn een veld te vinden dat overeenkwam met deze tekenketen. De komma is in dit geval niet nodig, maar het is een goede gewoonte om de verschillende mogelijkheden voor overeenkomende tekenketens van elkaar te scheiden met een teken, dat zeker niet overeenkomt.

Met de logische operatoren – .AND., .OR. en .NOT. – kunt u de computer precies vertellen, welke voorwaarden u wilt toepassen op de commando's. U kunt de computer snel de databaserecords laten doorzoeken om er bepaalde op te sporen, waarin u geïnteresseerd bent. Later in dit boek zullen we procedures bespreken, waarmee u de computer nog meer voor u kunt laten doen. De logische operatoren zullen nog belangrijker worden, wanneer u de computer begint aan te geven, wat u wilt dat hij automatisch doet.

Deel 4

DEEL VIER

Deel Vier gaat over kracht, snelheid en gemak. De mogelijkheden van de computer kunnen aanzienlijk worden vergroot door hem een paar nieuwe foefjes te leren. Dit is niet moeilijk – en het is werkelijk fantastisch, hoe goed u die foefjes kunt aanpassen bij wat u zelf nodig hebt op het gebied van informatieopslag en verslaggeving.

We gebruiken weer vertrouwde databasevoorbeelden om: keuzemogelijkheden in menu's te definiëren, 'do-while' instructies te laten werken en routine-werkzaamheden te automatiseren, waarmee we maatwerk krijgen en tijd en moeite besparen. Het maatwerk maken levert nog iets heel belangrijks: de mogelijkheid rapporten te laten vervaardigen, die geheel zijn aangepast aan onze eigen behoeften.

11. DE KUNST VAN HET PROCEDURESCHRIJVEN

De meeste database management systemen beschikken over een Query Language Processor en een Rapportenschrijver. Deze twee onderdelen kunnen waarschijnlijk de meeste, zo niet alle, behoeften vervullen, die u hebt. Alles wat u verder nog wenst, kan gemakkelijk verkregen worden met gebruikmaking van eenvoudige procedures. In voorgaande hoofdstukken hebben we een aantal voorbeelden besproken van bijzondere dingen die met procedures gedaan werden. Er zijn vele zaken die u kunt doen door met procedures precies aan te geven, wat de computer moet uitvoeren. Hiermee maakt u niet alleen computeruitvoer die voor u luxueus op maat is ontworpen, maar ook is dit een heel handige manier om u werk te besparen.

Om toe te lichten, hoe een procedure u moeite kan besparen, gaan we een eenvoudig voorbeeld doorwerken over een betaalkaartendatabase. Deze voorbeelddatabase, B:BTLKRTL, heeft het schema dat te zien is in Figuur 11-1.

Veld	Beschrijving	Veldnaam	Type	Breedte	Decimalen
1	Kaartnummer	NUMMER	N	4	
2	Begunstigde	BGNSTGDE	C	20	
3	Bedrag	BEDRAG	N	7	2
4	Storting of	STORTING	L	1	
5	Aftrekbaar?	AFTREKBR	L	1	
6	Al geboekt?	ALGEBOEKT	L	1	
7	Datum	DATUM	C	8	

Figuur 11-1 Databaseplan voor betaalkaarten en stortingen

In dit voorbeeld hebben we gekozen voor een logisch veld, dat aangeeft of het bedrag van een mutatie betrekking heeft op een storting (of bijboeking) dan wel op een uitgegeven betaalkaart. Een 'Y' geeft een storting aan.

Om te berekenen hoeveel er op de rekening staat, voeren we de vraagtaal-dialoog in Scherm 11-1.

```
.SUM BEDRAG TO ONTVANG FOR STORTING
10677.80
.SUM BEDRAG TO UITGAAF FOR.NOT.STORTING
9450.51
.STORE ONTVANG-UITGAAF TO SALDO
1227.29
```

Scherm 11-1

Merkt u op, dat de geheugenvariabele waarin de som van de stortingen wordt opgeslagen ONTVANG heet. Helaas bestaat de neiging die variabele STORTING te noemen. Een gegevensveld en een geheugenvariabele mogen echter nooit dezelfde naam hebben. Daar zou de computer van in de war raken.

Merk verder op dat we alle uitgeschreven betaalkaarten en alle stortingen iedere keer opnieuw optellen als we het saldo berekenen. In een conventioneel overzicht houden we het saldo doorgaans voortdurend bij. Met de computer is het vaak gemakkelijker het saldo telkens opnieuw uit te rekenen dan het steeds bij te houden.

Zoals u kunt zien, zijn drie instructies nodig om te bepalen, hoeveel op de rekening staat. Het laat zich aanzien, dat we vaak zullen willen weten, wat het saldo is. Om ons het werk van het iedere keer intypen van de drie instructies te besparen, zullen we iets maken, waarmee de computer zich de instructies 'herinnert'. Dat geheugensteuntje voor de computer is een 'procedure'.

De computer 'onthoudt' een dergelijke procedure door de instructies in een speciaal soort file te zetten. Die file wordt op een diskette gezet en staat steeds tot uw beschikking als u hem wilt gebruiken. Bij dBASE II heet deze speciale file COMMANDOFILE. Een commandofile maakt het mogelijk een groep commando's te 'bewaren', zodat we ze – als groep – kunnen gebruiken, zonder dat we ze elke keer geheel hoeven uit te typen.

Om de computer zich een procedure te laten 'herinneren' moet u hem eerst vertellen, dat u er een gaat schrijven. Bij dBASE II gebeurt dit met het commando MODIFY COMMAND.

```
.MODIFY COMMAND
```

De computer zal hierop antwoorden met een verzoek om een filenaam. Voor deze filenamen zijn dezelfde regels van toepassing als bij databasefiles en rapportagefiles. Een filenaam moet uit acht of minder letters en cijfers bestaan en moet met een letter beginnen. U moet ook door het plaatsen van een diskettedrive-aanduiding aangeven, in welke diskettedrive de procedure opgeslagen moet worden. In dit voorbeeld gaan we de commandofile SALDO noemen en zetten we hem in drive B.

ENTER FILE NAME: **B:SALDO**

Het scherm zal nu gewist worden. Op het scherm zal heel even te zien zijn

NEW FILE

De woorden 'NEW FILE' zullen verdwijnen en het scherm zal helemaal blanco zijn, op de cursor na, die in de linkerbovenhoek van het scherm staat. Er is geen puntaansporing. Niet dat er iets mis is – zo heeft de ontwerper dit commando nou eenmaal in elkaar gezet.

Typt u nu de instructies na elkaar in, net alsof u op een blanco vel papier aan het typen was. De instructies die u typt, verschijnen als afgebeeld in Scherm 11-2.

Merkt u op, dat aan het einde van de procedure een extra regel is toegevoegd. Die regel bestaat enkel uit het woord CANCEL. Dit woord dient om de besturing van de computer 'terug te geven' aan het toetsenbord nadat de computer de procedure heeft voltooid.

```
SUM BEDRAG TO ONTVANG FOR STORTING
SUM BEDRAG TO UITGAAF FOR.NOT.STORTING
STORE ONTVANG-UITGAAF TO SALDO
CANCEL
```

Scherm 11-2

De computer zal de instructies niet uitvoeren als u een procedure aan het schrijven bent. Omdat de meesten van ons niet foutloos typen, verschaft het systeem enkele mogelijkheden tot verbeteren via de control toets. De wijzi-

gingsmogelijkheden en de bijbehorende control toetsen zijn te zien in Figuur 11-2.

Controltoets	Uitwerking
S	Verplaatst de cursor 1 teken naar links
D	Verplaatst de cursor 1 teken naar rechts
E	Verplaatst de cursor 1 regel omhoog
X	Verplaatst de cursor 1 regel omlaag
N	Maakt een blanco regel
T	Verwijdert een regel
Y	Wist de inhoud van een regel
G	Wist een teken
V	Voegt tekens tussen
W	Schrijft de procedure weg naar diskette en laat het toetsenbord weer normaal werken
Q	Doet de procedure weg en laat het toetsenbord weer normaal werken

Figuur 11-2 Wijzigingsmogelijkheden bij MODIFY COMMAND

Wanneer de commando's zijn ingevoerd als afgebeeld in Scherm 11-2, houdt u eenvoudigweg de CONTROL toets ingedrukt, terwijl u de W aanslaat. De COMMANDOFILE, 'SALDO' genaamd, wordt dan naar diskette weggeschreven: u kunt dit commando nu gebruiken, zo vaak u maar wilt.

Dat gaat als volgt: als de betaalkaartendatabase in gebruik is en u de computer de procedure B:SALDO wilt laten uitvoeren, typt u dan 'DO' gevolgd door B:SALDO, als afgebeeld in Scherm 11-3. Merkt u op, dat de uitkomst van elke bewerking wordt weergegeven – de instructie zelf niet. Dit is zonder meer gemakkelijker dan de instructies iedere keer opnieuw typen, als we het saldo willen weten.

```
.DO B:SALDO
10677.80
9450.51
1227.29
DO CANCELLED
```

Scherm 11-3

Dit is geweldig. We hoeven maar een paar woorden te typen (DO B:SALDO) en het antwoord verschijnt. Maar misschien is er wel een klein probleempje: hoe kunnen we onthouden, welk getal wat voorstelt? Dat is ook gemakkelijk te regelen: we laten de computer gewoon niet alleen de antwoorden weergeven, maar ook de instructies, net als in Scherm 11-1. Gebruik hiervoor 'SET ECHO ON'. In het algemeen worden instructies dus niet weergegeven, maar in dit geval is het beter van wel.

Hiertoe voegen we 'SET ECHO ON' in bij onze bestaande procedure B:SALDO door gebruik te maken van het commando 'MODIFY COMMAND'.

.MODIFY COMMAND
ENTER FILE NAME:**B:SALDO**

Het scherm zal gewist worden en de computer zal de bestaande file B:SALDO weergeven. De cursor zal in de linkerbovenhoek van het scherm staan, precies op de 'S' van de eerste SUM. Het commando SET ECHO ON wordt op de eerste regel gezet door te drukken op CONTROL N – waardoor u een blanco regel krijgt – en vervolgens te typen SET ECHO ON. Gebruik CONTROL X om de cursor te verplaatsen naar de eerste C van het woord CANCEL. Druk op CONTROL N, waardoor u een blanco regel krijgt, en typt u vervolgens SET ECHO OFF in (de normale toestand). Het scherm moet er nu uitzien als Scherm 11-4.

```
SET ECHO ON
SUM BEDRAG TO ONTVANG FOR STORTING
SUM BEDRAG TO UITGAAF FOR.NOT.STORTING
STORE ONTVANG-UITGAAF TO SALDO
SET ECHO OFF
CANCEL
```

Scherm 11-4

Druk op CONTROL W om de nieuwe commandofile SALDO weg te schrijven naar diskettedrive B. Telkens wanneer u nu de computer de pro-

cedure B:SALDO laat uitvoeren, zullen de uitkomsten verschijnen als in Scherm 11-5.

```
.SUM BEDRAG TO ONTVANG FOR STORTING
10677.80
.SUM BEDRAG TO UITGAAF FOR.NOT.STORTING
9450.51
.STORE ONTVANG-UITGAAF TO SALDO
1227.29
```

Scherm 11-5

Het schrijven van een procedure gaat net zo als het schrijven van een brief. Het verschil is natuurlijk dat een procedure op de computer geschreven wordt in een taal die de computer verstaat. De meeste database management systemen voor microcomputers stellen u in de gelegenheid procedures te schrijven op twee verschillende manieren: via een interne functie van het DBMS of door gebruikmaking van een tekstverwerkingssysteem buiten het DBMS.

Een tekstverwerkingssysteem is een speciaal stuk programmatuur dat speciaal is ontworpen voor het werken met tekst.

Wanneer u een tekstverwerkingssysteem gebruikt om een procedure te schrijven, moet u behalve een filenaam ook de FILESOORT intypen. Een FILESOORT bestaat uit een punt gevolgd door drie letters. Wanneer u 'binnen' dBASE II werkt, wordt deze filesoort automatisch aan de naam toegevoegd – dBASE II 'weet' wat voor soort file u maakt, uit de commando's die u het systeem geeft, en voegt als filesoort .CMD voor u toe. Wanneer u via een tekstverwerkingssysteem werkt, moet de filesoort ingetypt worden als deel van de naam. Het tekstverwerkingssysteem weet namelijk niets over soorten files.

Bij dBASE II is de filesoort voor procedures (commandofiles) '.CMD'. Dankzij deze FILESOORT weet het database systeem dat de file een procedure is. Onze voorbeeldprocedure (commandofile) zou de naam

B:SALDO.CMD

moeten krijgen, wanneer deze op een tekstverwerkingssysteem aangemaakt zou worden.

Zelfs al kan de procedure binnen het database systeem geschreven worden, dan is het toch vaak beter deze te schrijven met een tekstverwerkingssys-

teem. Tekstverwerkingssystemen bieden vaak aanmerkelijk betere mogelijkheden tot wijzigen dan een DBMS. Daar zijn tekstverwerkingssystemen tenslotte voor. Deze wijzigingsmogelijkheden zijn erg nuttig, wanneer u lange procedures schrijft.

Tekstverwerkingssystemen hebben vaak een afzonderlijke werkstand voor het vervaardigen van materiaal dat door de computer gelezen moet worden. Alle procedures, commandofiles en computerprogramma's die op een tekstverwerkingssysteem aangemaakt worden, moeten in deze werkstand geschreven worden. De gewone werkstand voor tekstverwerking bij het vervaardigen van brieven en documenten voegt symbolen aan de tekst toe, die u niet kunt zien. Deze dienen voor intern gebruik bij het wijzigen en vervolgens afdrukken van de brieven en documenten. Deze extra symbolen kunnen problemen veroorzaken, wanneer ze opgenomen worden in een procedure. Als u niet zeker weet, welke werkstand u moet gebruiken, raadpleegt u dan de handleiding van het tekstverwerkingssysteem dat u gebruikt.

BESPAAR WAT TIJD EN ERGERNIS

Wanneer de computer een instructie als SUM uitvoert, moet hij de hele database van diskette 'lezen'. De tijd die het kost de database te lezen, hangt voornamelijk af van de grootte van de database en van het soort schijfdrive. De gemiddelde diskettedrive kan de database lezen met een snelheid van 1600 tekens per seconde. Het lezen van een heel kleine database kost zo iedere keer 10 seconden. Het lezen van een van 160 000 tekens kost 100 seconden.

Als u commando's met de hand invoert, zult u de korte tijd die de computer nodig heeft om de database te lezen erg hinderlijk gaan vinden, zodra die boven de vijf seconden uitkomt. Vanwege deze 'leestijd' is het vaak beter commando's bij elkaar in een procedure te zetten. Daarmee bevrijdt u uzelf van de hinderlijke korte wachttijden van een paar tot meerdere seconden bij elke instructie.

De gedachte achter het opzetten van een procedure is dat u de computer al het werk (of althans het meeste) laat doen. U geeft een lijst met instructies – hij voert ze allemaal achter elkaar uit. U regelt het. Hij is de trouwe bediende, die nooit klaagt. Als u veel achter het toetsenbord zit om de computer te ondervragen over de inhoud van uw database, kunt u uzelf heel wat tijd besparen met een procedure, die de computer laat doen, wat u wilt.

Stel bijvoorbeeld eens, dat u een heleboel informatie wilt hebben over sherry en cola uit de inventarisdatabase van onze slijterij. U kunt achter het toetsenbord plaatsnemen en een reeks commando's invoeren – OF u kunt

een procedure schrijven die een lijst is van die commando's en de computer zal het invoeren voor u doen.

```
COUNT FOR DRANK = 'SHERRY'
COUNT FOR DRANK = 'COLA'
COUNT FOR DRANK = 'SHERRY'.AND.INHOUD = '0.8 L'
enzovoorts.
```

Als u deze commando's vanaf het toetsenbord invoert, heeft de computer een paar seconden nodig voor het beantwoorden van elk ervan. Als de database groot is, kan elk antwoord vele seconden op zich laten wachten.

Wanneer u nu de commando's als procedure invoert, kunt u rustig een kop koffie gaan drinken, terwijl de computer de door u gewenste antwoorden opzoekt. De tijd die dat kost, is er wellicht niet minder om, maar terwijl de computer bezig is, zit u niet duimen te draaien.

We zijn allemaal in aanraking geweest met een of andere verzameling richtlijnen die exact stuk voor stuk moeten worden opgevolgd. Dit soort instructieverzamelingen gebruikt de computer ook. Om de computer de procedure te laten uitvoeren, moeten we de instructies zo opschrijven, dat hij ze kan begrijpen. Denkt u eraan, dat de computer wel heel erg snel kan zijn, maar dat hij niet erg slim is. Een computerprocedure moet heel duidelijk geschreven worden. U kunt niet aannemen dat de computer iets wel 'weet'.

U moet het volgende voorbeeld maar eens bekijken. Als we een heel klein kind vragen tot drie te tellen, dan zal dat kind daar waarschijnlijk wel in slagen. Als u de computer tot drie wilt laten tellen, dan is dat andere koek. De computer kan tot drie tellen als we hem 'leren' – een procedure geven – om tot drie te tellen. Zelfs als hij dat een keer gedaan heeft, zal hij niet in staat zijn het opnieuw te doen, tenzij de procedure opnieuw gebruikt wordt.

Om dit nader toe te lichten staat hieronder een gewone Nederlandstalige versie van een procedure om tot drie te tellen. Deze procedure lijkt erg op de procedure die een mens met een rekenmachine met geheugen zou volgen.

Stap 1. Sla een 'nul' op in het geheugen

Stap 2. Tel 1 op bij de inhoud van het geheugen en sla de uitkomst op in het geheugen

Stap 3. Toon de inhoud van het geheugen

Stap 4. Tel 1 op bij de inhoud van het geheugen en sla de uitkomst op in het geheugen

Stap 5. Toon de inhoud van het geheugen

Stap 6. Tel 1 op bij de inhoud van het geheugen en sla de uitkomst op in het geheugen

Stap 7. Toon de inhoud van het geheugen

Wanneer we willen dat de computer deze procedure uitvoert, moet hij anders geschreven worden. De vertaling van ons Nederlands in een computertaal zal anders zijn voor elke computertaal, net als die anders zou zijn bij vertaling in een andere menselijke taal als Engels of Frans. In de Application Development Language (ADL) van dBASE II ziet de vertaling er zo uit.

Stap 1. STORE 0 TO X

Stap 2 en 3. STORE X+1 TO X

Stap 4 en 5. STORE X+1 TO X

Stap 6 en 7. STORE X+1 TO X

De procedure zelf bevat slechts de tekst te beginnen met het woord STORE. De stapnummers staan er hier alleen bij om u de overeenkomst tussen de twee versies van de procedure te laten zien.

Uit deze voorbeeldprocedure kunt u twee dingen leren.

- Ten eerste, we doen een paar keer hetzelfde achter elkaar.
- Ten tweede, deze benadering is niet erg handig om een groot aantal malen te doen – bijvoorbeeld om tot duizend te tellen.

De Nederlandse versie kan herschreven worden als:

Stap 1. Sla een 'nul' op in het geheugen

Stap 2. Tel 1 op bij de inhoud van het geheugen en sla de uitkomst op in het geheugen

Stap 3. Toon de inhoud van het geheugen

Stap 4. Herhaal stap 2

Stap 5. Herhaal stap 3

Stap 6. Herhaal stap 2

Stap 7. Herhaal stap 3

In het boek Winnie The Poo van A. A. Milne ontdekken Winnie en het varkentje Piglet voetstappen in de sneeuw voor het huis van het varkentje. Dromend van grote avonturen gaan de twee de sporen volgen om te zien, waarheen ze leiden en van wat voor soort wezen ze afkomstig zijn. Ze volgen deze voetsporen een hele tijd, totdat ze tenslotte terug blijken te komen bij het huis van het varkentje. Daar ontdekken ze drie stel voetsporen die zich verwijderen van het huis van het varkentje. Het ene stel is veel kleiner dan de andere twee. Na enig redetwisten komen ze tot de speculatie dat ze een woozle en een wizzle op het spoor zijn. Opnieuw gaan ze de voetsporen achterna. Na een tijdje ontdekken ze dat bij de eerste drie stel voetstappen er nog twee zijn gekomen. Het varkentje begint zich erge zorgen te maken over zijn veiligheid en bedenkt dat hij thuis nog wat te doen heeft. Hij laat Winnie alleen. Winnie de Poo komt er tenslotte achter, dat de voetstappen van henzelf zijn....ze hebben rondjes gelopen.

In het geval van Winnie de Poo en Piglet is het rondjes lopen misschien wel een groot avontuur geweest, maar in feite hebben ze niets bereikt. In ons geval blijkt echter dat in een kringetje lopen ons vooruitbrengt. Kringetjes maken het schrijven van eenvoudige procedures mogelijk.

DO WHILE...ENDDO

Vereenvoudiging # = 1

Stap 1. Sla een 'nul' op in het geheugen

Stap 2. Tel 1 op bij de inhoud van het geheugen en sla de uitkomst op in het geheugen

Stap 3. Toon de inhoud van het geheugen

Stap 4. Als het geheugen kleiner is dan 3 ga dan naar stap 2

Vereenvoudiging # = 2

Stap 1. Sla een 'nul' op in het geheugen

Stap 2. Voer de stappen 3 en 4 uit zolang de inhoud van het geheugen kleiner is dan 3

Stap 3. Tel 1 op bij de inhoud van het geheugen en sla de uitkomst op in het geheugen

Stap 4. Toon de inhoud van het geheugen

Het eerste voorbeeld is kenmerkend voor de manier waarop u een procedure zou kunnen schrijven in een van de 'traditionele' computertalen als FORTRAN, COBOL of BASIC.

Het tweede voorbeeld vertegenwoordigt de 'moderne' talen als PASCAL, PL/1 en ADL. Als we ons eenvoudige telvoorbeeld (Vereenvoudiging # =2) in ADL schrijven, krijgen we:

```
STORE 0 TO X
DO WHILE X<3
STORE X+1 TO X
ENDDO
```

Het pijlpuntje '<' is de rekenkundige afkorting die 'kleiner dan' betekent. De instructie die met DO begint, houdt in 'doe het volgende zolang X kleiner is dan 3'. Het pijlpuntje andersom ('>') betekent 'groter dan'. De instructie zou geschreven kunnen worden als 'DO WHILE 3>X', wat inhoudt 'doe het volgende zolang 3 groter is dan X'. De twee instructies DO WHILE X<3 en DO WHILE 3>X betekenen precies hetzelfde.

Dit nieuwe voorbeeld heeft net zoveel instructies als het oorspronkelijke. Er is echter een belangrijk verschil – we kunnen de computer tot honderd of tot duizend of tot een miljoen laten tellen door enkel de '3' door het gewenste eindgetal te vervangen. Het commando DO WHILE X<3 vertelt de computer dat u wilt dat hij doorgaat met herhalen van de volgende instructies, zolang de waarde van X kleiner is dan drie. ENDDO geeft het einde aan van de reeks instructies, die bij DO WHILE begonnen is. Elke DO WHILE moet een ENDDO hebben. DO WHILE – ENDDO is een van de manieren om de computer hetzelfde stel instructies telkens opnieuw te laten uitvoeren, zolang aan een of andere voorwaarde [zoals X<3] voldaan is. De groep instructies die begint met DO WHILE en eindigt met ENDDO heet een 'LUS'.

DE LUS INITIALISEREN

Onmiddellijk voorafgaand aan de lus hebben we een instructie gebruikt, die de waarde 0 opsloeg in de geheugenvariable X. Als u tot drie telt, weet u, dat u bij één moet beginnen. De computer weet niet, waar hij moet beginnen. Hij moet eerst te horen krijgen, waar hij moet beginnen – net zoals hij te horen moet krijgen, hoe hij moet tellen. Deze enkele instructie – sla 0 op in X – doet twee dingen:

- De waarde 0 wordt opgeslagen in X
- Eerst wordt ook nog de geheugenvariabele X 'gemaakt'

U mag in een procedure een geheugenvariabele niet gebruiken voordat die variabele is 'gemaakt'. Het is u ook niet toegestaan de geheugenvariabele te 'maken' zonder er een beginwaarde aan toe te kennen.

In dit geval werd de variabele gemaakt en kreeg hij een beginwaarde met de instructie 'STORE 0 TO X'. Deze instructie 'initialiseert' de LUS door te zorgen voor het beginpunt en voor het maken van de variabele X.

TELLER: EEN HOOFDBEGRIP

Deze eenvoudige procedure die kan tellen, is een TELLER. De teller vormt het uitgangspunt van de gewone telmachine en van de elektronische zakrekenmachine. De teller is een hoofdbegrip dat vaak wordt gebruikt in procedures bij zakelijke toepassingen van database systemen.

Procedures die ingewikkelder taken vervullen, bestaan meestal uit een groep eenvoudige procedures. Stel bijvoorbeeld eens, dat u in stappen van 1 tot tien wilt tellen en daarna in stappen van 10 tot honderd. Eén manier om dit te doen is het gebruik van twee van onze eenvoudige tellussen na elkaar.

```
STORE 0 TO X

DO WHILE X < 10
STORE X + 1 TO X
ENDDO

DO WHILE X < 100
STORE X + 10 TO X
ENDDO
```

EEN ALTERNATIEVE PROCEDURE: 'IF'

Nog een manier om precies hetzelfde resultaat te bereiken is de computer voor verschillende waarden van X verschillende dingen te laten doen. Dat gebeurt hierboven natuurlijk ook, alleen maakt deze procedure gebruik van het feit, dat we alles van X weten en ook alles weten van wat we willen dat er gebeurt. De computer is in staat beslissingen te nemen – al zijn het nogal beperkte beslissingen. We kunnen daarvan gebruik maken en een gelijkwaardige procedure schrijven.

```
STORE 0 TO X

DO WHILE X < 100
IF X < 10
   STORE X + 1 TO X
ENDIF

IF X > = 10
   STORE X + 10 TO X
ENDIF

ENDDO
```

'IF' is het woord dat we gebruiken als we de computer een beslissing willen laten nemen over het wel of niet doen van iets. Het wordt precies zo gebruikt als we normaal het woord ALS gebruiken. ALS het regent, steken we een paraplu op; ALS de benzinetank bijna leeg is, moeten we stoppen om te tanken. We gebruiken ALS, wanneer hetgeen gedaan moet worden (of de conclusie die getrokken moet worden) van een of andere voorwaarde afhangt.

In dit voorbeeld is het optellen van het ene getal bij het andere hetgeen gedaan moet worden. De voorwaarde is het vergelijken van de teller met 10. Bij het nemen van een van beide beslissingen weet de computer niets af van de andere IF. Bij elke IF moet een ENDIF staan, net zoals bij elke DO WHILE en ENDDO moet staan. De informatie na IF (X < 10) is de voorwaarde waarover de computer een beslissing moet nemen. De beslissing die de computer neemt, is of X al dan niet kleiner dan 10 is. Zo ja, dan gaat de IF op en zal de computer de instructie STORE X+1 to X uitvoeren. Zo nee, dan zal de computer niet X+1 in X opslaan.

DO WHILE en IF zijn de belangrijkste gereedschappen die gebruikt kunnen worden bij alle procedures die u ooit nodig zult hebben. Bij deze voorbeelden gebruiken we de eigen terminologie van de Application Development Language van dBASE II. De grondbegrippen zijn echter algemeen en worden bij alle computertalen gebruikt. De terminologie is dezelfde als die van moderne talen als PL/1 en PASCAL.

Om u een betere indruk te geven van hoe u DO WHILE en IF kunt gebruiken, gaan we met deze twee voorzieningen een procedure schrijven, die hetzelfde doet als B:SALDO. We nemen deze gelegenheid ook te baat om te laten zien, hoe u de schermbeelden van de computer beter kunt beïnvloeden. De nieuwe B:SALDO is te zien in Scherm 11-6. De oorspronkelijke uitkomsten van de procedure (Scherm 11-1) worden hier voor uw gemak nogmaals weergegeven.

```
.SUM BEDRAG TO ONTVANG FOR STORTING
10677.80
.SUM BEDRAG TO UITGAAF FOR.NOT.STORTING
9450.51
.STORE ONTVANG-UITGAAF TO SALDO
1227.29
```

Scherm 11-1

```
USE B:BTLKRTL
SET TALK OFF
STORE 0 TO ONTVANG, UITGAAF
DO WHILE .NOT.EOF
IF STORTING
STORE BEDRAG + ONTVANG TO ONTVANG
ENDIF
IF.NOT.STORTING
STORE BEDRAG + UITGAAF TO UITGAAF
ENDIF
SKIP
ENDDO
? 'TOTAAL STORTINGEN', ONTVANG
? 'TOTAAL BETAALKAARTEN', UITGAAF
? 'SALDO ', ONTVANG-UITGAAF
SET TALK ON
CANCEL
```

Scherm 11-6

In ons voorbeeld gebruiken we instructies die u misschien niet begrijpt:

```
SET TALK OFF/ON
SKIP
DO WHILE.NOT.EOF
?
```

SET TALK OFF/ON

Wellicht is het u opgevallen, dat veel commando's – bijvoorbeeld STORE, SUM – iedere keer dat ze gebruikt worden, een antwoord laten zien. Dit is één van de manieren, waarop de computer tegen u 'praat'. Dit is handig, wanneer u vanaf het toetsenbord met uw computer werkt. Het is vaak niet zo handig, wanneer u procedures gebruikt. Het scherm (en/of de printer) loopt vol met dit soort antwoorden. Bij dBASE II kan het zichtbare antwoord op een commando aan- en uitgezet worden. SET TALK ON en SET TALK OFF zijn de commando's waarmee u dat doet.

SKIP

Het database management systeem werkt in feite met maar één record tegelijk. Wanneer de USE instructie wordt gegeven, wordt het DBMS bovenaan de database gezet – op record 1. Elke keer dat de opdracht SKIP wordt gegeven gaat het DBMS één record verder. Wanneer het laatste record wordt bereikt, zal de volgende SKIP het DBMS erop wijzen, dat het einde van de file is bereikt.

DO WHILE.NOT.EOF

Zoals we in vorige voorbeelden hebben gezien, is het commando DO WHILE van toepassing zolang aan een of andere voorwaarde is voldaan. EOF wordt uitgesproken als 'end of file', einde van een file. Het commando betekent letterlijk 'computer – doe het volgende steeds opnieuw, totdat je bij het einde van de database bent gekomen'. Waarschijnlijk is dit de meestgebruikte vorm van DO WHILE. Hiermee zal automatisch de DO LUS beëindigd worden, wanneer het einde van de database bereikt wordt.

?

Het vraagteken is een veelzijdig commando dat het mogelijk maakt bepaalde informatie te laten weergeven. De enkele aanhalingstekens aan weerszijden van de tekst – bijvoorbeeld bij 'TOTAAL STORTINGEN ' – heten scheidingstekens. De aanwezigheid ervan geeft aan dat het omslotene tekst is, die weergegeven moet worden. De naam van de geheugenvariabele geeft aan, dat de inhoud van die geheugenvariabele weergegeven moet worden. De komma wordt gebruikt om de verschillende weer te geven zaken van elkaar te scheiden. Elk vraagteken zal zorgen voor een nieuwe regel.

Onze nieuwe voorbeeldprocedure B:SALDO is een klein beetje ingewikkelder dan onze oorspronkelijke versie. Maar het beetje werk heeft gezorgd voor een resultaat dat veel beter aansluit bij onze behoeften. Wanneer we de computer deze commandofile laten uitvoeren, zal het schermbeeld van Scherm 11-7 verschijnen.

```
.DO B:SALDO
TOTAAL STORTINGEN          10677.80
TOTAAL BETAALKAARTEN        9450.51
SALDO                       1227.29
DO CANCELLED***
```

Scherm 11-7

Met procedures kan de computer meer werk voor u doen. In dit hoofdstuk hebben we een paar hoofdbegrippen aan u voorgesteld: de teller, de DO lus, de voorwaarde (IF) evenals een paar dBASE II commando's die vaak

gebruikt worden in procedures. Procedures zoals die in dit hoofdstuk zijn leuk om te schrijven en besparen u uiteindelijk tijd en moeite. Bovendien maken ze de kans op fouten kleiner. Als een procedure eenmaal foutloos geschreven is, kan de computer hem telkens opnieuw zonder fouten uitvoeren.

12. EEN PROCEDURE AAN HET WERK ZETTEN

Procedures zijn bedoeld om u tijd, moeite en geld te besparen en ook om een meer bevredigend resultaat te bereiken. Ze kunnen gebruikt worden voor bijna alles wat u maar kunt bedenken. Het 'automatiseren' van routinewerk is één van de nuttige dingen waar de computer geschikt voor is. De computer is er tenslotte om u het leven gemakkelijker te maken.

In dit hoofdstuk gaan we een nogal uitgebreid voorbeeld bekijken van wat een computer plus een database management systeem kan doen. We blijven nog steeds op voor iedereen vertrouwd terrein en gaan een overzicht van girobetaalkaarten en stortingen (of bijboekingen) 'op de computer zetten'. Wat we doen met het betaalkaartenoverzicht is niet alleen vertrouwd: we kunnen er een aantal nuttige ideeën mee laten zien.

Deze voorbeelddatabase, B:BTLKRTL, heeft de opzet die te zien is in Figuur 12-1.

Veld	Beschrijving	Veldnaam	Type	Breedte	Decimalen
1	Kaartnummer	NUMMER	N	4	
2	Begunstigde	BGNSTGDE	C	20	
3	Bedrag	BEDRAG	N	7	2
4	Storting of	STORTING	L	1	
5	Aftrekbaar?	AFTREKBR	L	1	
6	Al geboekt?	ALGEBOEKT	L	1	
7	Datum	DATUM	C	8	

Figuur 12-1 Databaseplan voor betaalkaarten en stortingen

De velden in dit plan komen overeen met die in een gewoon giro-overzicht op één uitzondering na: er is geen veld om steeds het saldo bij te houden. Bij een klassiek overzicht is het van groot belang dat steeds bij te houden. Het is daarbij duidelijk niet uitvoerbaar het saldo iedere keer opnieuw uit te rekenen als we het gebruiken. Bij gebruik van de computer gaat dat echter niet op, omdat het zeer wel mogelijk is alle berekeningen te laten doen bij iedere keer dat we een kaart uitschrijven. Dit hebben we al in Hoofdstuk 11 laten zien.

Om een dergelijk overzicht bij te houden, moet u een aantal routinewerkzaamheden steeds uitvoeren. Bijvoorbeeld:

1 Een betaalkaart bijschrijven
2 Een storting bijschrijven
3 Iets veranderen om een fout te verbeteren
4 Kijken of u een bepaalde betaalkaart hebt afgegeven
5 Aftrekbare uitgaven bij elkaar zetten voor uw belastingaangifte
6 Stand van uw rekening bekijken, saldo
7 Vergelijken met de giro-afrekening

Een aantal van deze dingen kunt u doen, 'wanneer ze nodig zijn'. Andere dingen doet u regelmatig, bijvoorbeeld dagelijks, maandelijks of jaarlijks. Alles kan vanaf het toetsenbord van uw computer gedaan worden met gebruikmaking van de database en de vraagtaal. Maar het is doelmatiger – en u zult u meer op uw gemak voelen – als uw overzicht volgens vaste procedures wordt bijgehouden. De procedure legt een goede werkwijze vast, en u hoeft u verder niet druk te maken.

In Hoofdstuk 11 hebben we de grondgedachten van een procedure besproken en ook hoe u er een kunt maken. In de rest van dit hoofdstuk gaan we die grondgedachten en die methoden gebruiken om een stel procedures te schrijven, die ons overzicht bijhouden. Dit stel procedures is bedoeld om de werking ervan te verduidelijken. Ze vormen geen complete giro-administratie.

Om ons doel te bereiken en ons giroverkeer op te tekenen hebben we in totaal zeven procedures nodig. Eén hebben we er al (nummer 6 – het bepalen van het saldo van het moment). Ze zijn alle zeven ongeveer even lang als ons voorbeeld B:SALDO. En omdat ze allemaal bij dezelfde bezigheid horen – het bijhouden van het giro-overzicht – is het goed er keuzemogelijkheden van te maken in een menu.

Het maken van een menu is nog eenvoudiger dan het maken van onze voorbeeldprocedure B:SALDO. Om dit snel en duidelijk te doen nemen we onze lijst van menuregels en maken we er rechtstreeks een menu van. De menuprocedure B:GIROMENU is te zien in Figuur 12-2.

```
USE B:BTLKRTL
INDEX ON NUMMER TO B:NUMMER
USE B:BTLKRTL INDEX B:NUMMER
SET TALK OFF
STORE 0 TO ONTVANG, UITGAAF
```

(vervolg op de volgende bladzijde)

```
DO WHILE T
ERASE
?
?
?'                          MENU BETAALKAARTENLIJST'
?
?'              1.    Invoeren betaalkaart'
?'              2.    Invoeren storting'
?'              3.    Veranderen ter herstel van een fout'
?'              4.    Kijken of een bepaalde kaart is afgegeven'
?'              5.    Lijst aftrekbare kaarten voor belastingaangifte'
?'              6.     Bepalen huidige saldo'
?'              7.    Vergelijken met giro-afrekening'
?'              8.    EINDE'
?
?'                SLA HET CIJFER VAN UW KEUZE AAN'
WAIT TO KEUZE
ERASE
IF KEUZE = '1'
   DO B:INVKRT
ENDIF
IF KEUZE = '2'
   DO B:STRTNGN
ENDIF
IF KEUZE = '3'
   DO B:VERANDER
ENDIF
IF KEUZE = '4'
   DO B:KIJKEN
ENDIF
IF KEUZE = '5'
   DO B:AFTREK
ENDIF
IF KEUZE = '6'
   DO B:SALDO
ENDIF
IF KEUZE = '7'
   DO B:CNTRLRN
ENDIF
IF KEUZE = '8'
   SET TALK ON
   CANCEL
ENDIF
ENDDO
```

Figuur 12-2 Menu betaalkaartenlijst

Opnieuw hebben we een paar nieuwe instructies toegepast. Dat zijn:

```
DO WHILE T
ERASE
WAIT TO KEUZE
?
```

DO WHILE T

Dit betekent letterlijk 'blijf steeds maar door doen'. De instructie DO WHILE betekent gewoonlijk doen, zolang de volgende voorwaarde waar is. In dit geval is 'T' (voor 'true', waar) altijd waar. Omdat we in werkelijkheid niet eeuwig deze lus willen blijven doorlopen, hebben we menuregel 8 opgenomen, die een ontsnappingsmogelijkheid biedt terug naar het database management systeem.

ERASE

Deze instructie wist alle aanwezige tekst van het beeldscherm.

WAIT TO KEUZE

Het commando WAIT heeft tot gevolg, dat een procedure tijdelijk ophoudt. De procedure zal verder gaan nadat u een willekeurig teken hebt aangeslagen.

WAIT TO KEUZE betekent dat het teken dat u aanslaat, opgeslagen moet worden in de geheugenvariabele 'KEUZE' en dat de procedure weer verder moet gaan. De inhoud van KEUZE wordt gebruikt om de computer duidelijk te maken, wat hij moet doen.

?

Het vraagteken kan op allerlei manieren worden gebruikt. Wanneer het in zijn eentje wordt gebruikt, zal het zorgen voor een blanco regel op het beeldscherm of op de printer. Het wordt ook toegepast om de inhoud van geheugenvariabelen en gegevensvelden te weergeven.

Tekst die is omgeven door dubbele of enkele aanhalingstekens zal als tekst op een regel verschijnen. Spaties kunnen gebruikt worden om de tekst op het beeldscherm of op de printer een bepaalde plaats te geven.

De menuprocedure zal als volgt werken: het invoeren van het commando DO B:GIROMENU zal ervoor zorgen, dat het menu op het scherm verschijnt als afgebeeld in Scherm 12-1.

```
                MENU BETAALKAARTENLIJST

        1.   Invoeren betaalkaart
        2.   Invoeren storting
        3.   Veranderen ter herstel van een fout
        4.   Kijken of een bepaalde kaart is afgegeven
        5.   Lijst aftrekbare kaarten voor belastingaangifte
        6.   Bepalen huidige saldo
        7.   Vergelijken met giro-afrekening
        8.   EINDE

     SLA HET CIJFER VAN UW KEUZE AAN

WAITING
```

Scherm 12-1

Druk de cijfertoets in, die met uw keuze overeenkomt. Dit is weer een geval waarin u niet op de RETURN toets hoeft te drukken. Het cijfer dat u aanslaat, wordt als teken opgeslagen in de geheugenvariabele KEUZE. De IF instructie die overeenkomt met uw keuze, zorgt ervoor dat de computer een procedure gaat uitvoeren. Als u bijvoorbeeld nummer 6 hebt uitgekozen, zal de computer de commandofile B:SALDO gaan uitvoeren.

Het eerste stel instructies in de menuprocedure (Figuur 12-2) zal ervoor zorgen, dat de database een index krijgt op het betaalkaartnummer. In werkelijkheid zou u niet telkens opnieuw een index laten maken, bij gebruik van dit menu. Het maken van de index staat hier alleen om duidelijk te maken, dat de procedure uitgaat van het gebruik van een geïndexeerde database.

MENUKEUZE #1

Het indrukken van de '1' toets zorgt voor het kiezen van menuregel 1, waarmee het mogelijk wordt een nieuwe betaalkaart in uw betaalkaartenlijst in de computer op te nemen. U zou dit natuurlijk ook kunnen doen via het commando APPEND, maar in dit voorbeeld willen we een manier aangeven om de resultaten van APPEND te bereiken met volledig verduidelijkende aansporingen voor ieder gegeven.

Eén mogelijke procedure waarmee u een nieuwe betaalkaart kunt invoeren, staat in Figuur 12-3. Opnieuw hebben we een paar nieuwe commando's gebruikt:

ACCEPT
INPUT
RETURN
APPEND BLANK

ACCEPT en INPUT

ACCEPT en INPUT geven de computer een mogelijkheid u te 'vragen' tijdens een procedure gegevens in te voeren vanaf het toetsenbord. INPUT wordt gebruikt om numerieke gegevens in te voeren. ACCEPT wordt gebruikt voor het invoeren van gegevens in tekenvorm. Voor de rest zijn de twee hetzelfde. De commandovorm wordt gedemonstreerd in de voorbeelden van Figuur 12-3. Het commando geeft de gewenste tekst weer, maakt een geheugenvariabele als MKAARTNO en wacht totdat u gegevens in de variabele invoert.

RETURN

RETURN is net zoiets als CANCEL: de procedure wordt erdoor beëindigd – in dit geval de procedure voor het invoeren van een nieuwe betaalkaart. Daarmee keert de besturing van de computer terug bij het menuprogramma B:GIROMENU, dat opnieuw het betaalkaartenlijstmenu weergeeft.

CANCEL zou aan alles een einde maken, waarbij de besturing van de computer terugkeert naar het toetsenbord.

APPEND BLANK

APPEND BLANK, dat vaak gebruikt wordt bij procedures, is een variatie op het commando APPEND. APPEND is de enige manier om records aan een database toe te voegen. APPEND BLANK voegt eenvoudigweg een blanco record toe, maar toont dat record niet voor het invoeren van gegevens. In dit voorbeeld wordt een blanco record gemaakt, vervolgens gegevens ingevoerd in geheugenvariabelen met ACCEPT en INPUT commando's en worden de gegevens uit de geheugenvariabelen overgebracht naar het blanco record door gebruik van het commando REPLACE. Het was niet nodig SET TALK OFF te gebruiken, omdat dit commando al eerder was gegeven door de menuprocedure B:GIROMENU (Figuur 12-2).

In dit voorbeeld verschijnen de aanwijzingen stuk voor stuk en worden de gegevens steeds met één instructie tegelijk ingevoerd. In dat opzicht is de procedure niet zo handig als APPEND, omdat de cursor niet kan worden teruggezet om het vorige gegeven te verbeteren.

```
APPEND BLANK
INPUT 'GEEF BETAALKAARTNUMMER' TO MKAARTNO
ACCEPT 'BETAALD AAN' TO MBGNSTGD
INPUT 'GEEF BEDRAG VAN DE KAART' TO MBEDRAG
ACCEPT 'GEEF DATUM (dd/mm/jj)' TO MDATUM
ACCEPT 'IS DEZE KAART AFTREKBAAR (Y/N)' TO MAFTREK
REPLACE NUMMER WITH MKAARTNO, BGNSTGDE WITH MBGNSTGD, BEDRAG;
    WITH MBEDRAG, DATUM WITH MDATUM, AFTREKBR WITH MAFTREK, STORTING WITH N
RETURN
```

Figuur 12-3 Voorlopige procedure voor invoer betaalkaart

Een schermbeeld dat lijkt op dat bij APPEND – met meer verduidelijkende aansporingen in plaats van de veldnamen – zou veel beter zijn. Om verder de gegevens meer ontspannen te kunnen invoeren zou het ook mogelijk moeten zijn de cursor een of meerdere velden terug te zetten om fouten daar alsnog te kunnen verbeteren. Bij dBASE II krijgt u dit voor elkaar door gebruik te maken van commando's die hier speciaal voor bestemd zijn.

De algemene vorm van het dBASE II commando is:

```
@ REGEL, KOLOM SAY 'WAT U MAAR WILT' GET VELDNAAM
READ
```

Het eerste commando, dat met '@', maakt het mogelijk de plaatsing van gegevens op het scherm te regelen. De meeste computerterminals hebben een beeldscherm met 24 regels van elk 80 tekens. De regels zijn van boven tot onder genummerd van 0 tot en met 23. De kolommen (tekenposities) zijn van links naar rechts genummerd van 0 tot en met 79. Om DIT IS EEN VOORBEELD te laten weergeven op regel 5, te beginnen op positie (in kolom) 10, is het commando

```
@ 5, 10 SAY 'DIT IS EEN VOORBEELD'
```

De tekst 'DIT IS EEN VOORBEELD' zal worden weergegeven op halve intensiteit net zoals met veldnamen gebeurt bij APPEND.

Evenzo kan de inhoud van een veld worden weergegeven met

```
@ 5, 10 GET BGNSTGDE
```

Dit voorbeeld geeft de huidige inhoud van het veld BEGNSTGDE weer op volle intensiteit – net zoals bij gebruik van het commando EDIT.

De twee voorbeelden kunnen worden samengevoegd tot

```
@ 5, 10 SAY 'DIT IS EEN VOORBEELD' GET BGNSTGDE
```

Het commando READ stelt u in staat de inhoud van het veld aangegeven door GET te veranderen. Een voorbeeldprocedure voor het invoeren van betaalkaarten is afgebeeld in Figuur 12-4. Het schermbeeld dat deze procedure voortbrengt, wordt getoond als Scherm 12-2.

```
APPEND BLANK
@ 5, 10 SAY 'GEEF BETAALKAARTNUMMER' GET NUMMER
@ 7, 10 SAY 'BETAALD AAN' GET BGNSTGDE
@ 9, 10 SAY 'GEEF BEDRAG VAN DE KAART' GET BEDRAG
@ 11, 10 SAY 'GEEF DATUM (dd/mm/jj)' GET DATUM
@ 13, 10 SAY 'IS DEZE KAART AFTREKBAAR (Y/N)' GET AFTREKBR
READ
REPLACE STORTING WITH N
RETURN
```

Figuur 12-4

Deze procedure zorgt ervoor dat de computer zich gedraagt op een manier die heel erg lijkt op APPEND behalve dat verduidelijkende aansporingen in plaats komen van de veldnamen. Alle CONTROL toetsen voor het verplaatsen van de cursor zijn dezelfde als bij APPEND. U ziet dat niet alle velden worden weergegeven – alleen de velden die voor deze procedure van belang zijn, zijn uitgekozen.

Het resultaat van deze procedure staat in Scherm 12-2. De cursor staat bij het begin op de plaats van het eerste teken van het veld NUMMER – gereed voor het ontvangen van gegevens.

```
GEEF BETAALKAARTNUMMER:        :
BETAALD AAN:                   :
GEEF BEDRAG VAN DE KAART:              :
GEEF DATUM (dd/mm/jj):             :
IS DEZE KAART AFTREKBAAR (Y/N):  :
```

Scherm 12-2

Omdat de kaartnummers meestal op volgorde zullen worden ingevoerd, kunt u zich wat moeite besparen door de computer het kaartnummer voor u te laten invoeren. De herziene procedure die dat doet, is afgebeeld in Figuur 12-5. De veranderingen zijn vet weergegeven.

```
GO BOTTOM
STORE NUMMER + 1 TO NP1
APPEND BLANK
REPLACE NUMMER WITH NP1, STORTING WITH N
@ 5, 10 SAY 'BETAALKAARTNUMMER' GET NUMMER
CLEAR GETS
@ 7, 10 SAY 'BETAALD AAN' GET BGNSTGDE
@ 9, 10 SAY 'GEEF BEDRAG VAN DE KAART' GET BEDRAG
@ 11, 10 SAY 'GEEF DATUM (dd/mm/jj)' GET DATUM
@ 13, 10 SAY 'IS DEZE KAART AFTREKBAAR (Y/N)' GET AFTREKBR
READ
RETURN
```

Figuur 12-5 Procedure voor invoer betaalkaarten B:INVKRT

Het commando GO BOTTOM zet het DBMS op het laatste record van de database. Dit record bevat het laatstgebruikte kaartnummer. Het nummer van de kaart die u wilt invoeren, moet het nummer zijn, dat daarna komt. Dit nieuwe kaartnummer wordt tijdelijk opgeslagen in de geheugenvariabele NP1 door STORE NUMMER + 1 TO NP1. Een blanco record wordt aan de database toegevoegd. Het nieuwe kaartnummer wordt in het blanco record geschreven door REPLACE NUMMER WITH NP1.

De instructie CLEAR GETS voorkomt dat de cursor gaat naar de velden met GET commando's , die voor CLEAR GETS staan. Het schermbeeld waarvoor deze procedure zorgt, lijkt op Scherm 12-2, maar de cursor staat op het meest linkse teken van het veld BGNSTGDE en het kaartnummer is ingevuld. Zie Scherm 12-3.

```
BETAALKAARTNUMMER: 25:
BETAALD AAN:                    :
GEEF BEDRAG VAN DE KAART:              :
GEEF DATUM (dd/mm/jj):                 :
IS DEZE KAART AFTREKBAAR (Y/N):  :
```

Scherm 12-3

MENUKEUZE # 2

Met procedure nummer 2 lunt u stortingen invoeren. Figuur 12-6 geeft deze procedure weer, genaamd B:STRTNGN.

```
APPEND BLANK
REPLACE STORTING WITH Y, BGNSTGDE WITH 'STORTING'
@ 9, 10 SAY 'GEEF BEDRAG VAN STORTING' GET BEDRAG
@ 11, 10 SAY 'GEEF DATUM (dd/mm/jj)' GET DATUM
READ
RETURN
```

Figuur 12-6. Procedure voor invoer vn een storting

Hij lijkt op B:INVKRT – de procedure voor het invoeren van betaalkaarten, maar is veel eenvoudiger. Scherm 12-4 geeft het schermbeeld dat met B:STRTNGN ontstaat.

```
GEEF BEDRAG VAN STORTING:        :
GEEF DATUM (dd/mm/jj):           :
```

Scherm 12-4

Opnieuw lijkt de procedure op APPEND behalve dat de verduidelijkende aansporingen extra zijn toegevoegd en dat niet alle velden worden weergegeven.

MENUKEUZE # 3

Het derde menu stelt u in staat 'fouten te verbeteren' in een eerder ingevoerd record. Bij dBASE II zou dit gedaan kunnen worden met een van de commando's EDIT en BROWSE. Zoals u nu wel zult verwachten, kunt u een procedure schrijven, die gelijk is aan EDIT, behalve dat de prompts verduidelijkende tekst zijn in plaats van veldnamen.

Een procedure die dit doet, is afgebeeld in Figuur 12-7.

```
ACCEPT 'DRUK OP K VOOR KAART, OP S VOOR STORTING' TO TYPE
IF TYPE = 'K'
   ACCEPT 'GEEF KAARTNUMMER' TO OPZOEK
   FIND &OPZOEK
   ERASE
   @ 5, 10 SAY 'BETAALKAARTNUMMER' GET NUMMER
   CLEAR GETS @ 7, 10 SAY 'BETAALD AAN' GET BGNSTGDE
   @ 9, 10 SAY 'GEEF BEDRAG VAN DE KAART' GET BEDRAG
   @ 11, 10 SAY 'GEEF DATUM (dd/mm/jj)' GET DATUM
   @ 13, 10 SAY 'IS DEZE KAART AFTREKBAAR (Y/N)' GET AFTREKBR
   READ
ENDIF
IF TYPE = 'S'
   ACCEPT 'GEEF DATUM VAN STORTING' TO OPZOEK
   DISPLAY FOR STORTING.AND.DATUM = OPZOEK
   INPUT 'GEEF TE WIJZIGEN RECORDNUMMER' TO OPZOEK
   GOTO OPZOEK
   ERASE
   @ 9, 10 SAY 'GEEF BEDRAG VAN STORTING' GET BEDRAG
   @ 11, 10 SAY 'GEEF DATUM (dd/mm/jj)' GET DATUM
   READ
ENDIF
RETURN
```

Figuur 12-7 Een procedure voor het wijzigen van een record

Ons eerste probleem is het vinden van het gewenste record. Als we de gegevens over een betaalkaart willen veranderen, dan is dat nogal eenvoudig – we laten de computer het record opzoeken, dat het kaartnummer bevat van de kaart waarvan we de gegevens willen veranderen. Als we echter een storting wensen, moeten we het record op een andere manier weten te vinden, omdat er in het veld NUMMER niets is ingevoerd, toen we de storting in de database opnamen.

In onze voorbeelddatabase is er geen uniek herkenningsteken voor stortingsrecords (behalve natuurlijk het recordnummer). Dit geeft weer eens het nut van planning aan. Ons gebrek aan planning heeft ervoor gezorgd dat we nu een noodoplossing moeten zoeken. Het is verleidelijk de datum te gebruiken als ons criterium voor de verandering. Zelfs als we nooit meer dan één storting per dag hebben, dan nog zouden we een fout kunnen maken bij het invoeren van een storting, waardoor er voor één dag wel meerdere stortingen in de database aanwezig zijn.

De oplossing is het recordnummer als criterium te gebruiken. Het laten weergeven van alle records voor een bepaalde dag maakt het juiste recordnummer toegankelijk.

De procedure bestaat uit twee delen, die gekozen worden al naar gelang het record dat we willen veranderen, een betaalkaart of een storting is.

1 Als het een betaalkaart is, vinden we het record met het gewenste kaartnummer en vervolgens laten we hetzelfde schermbeeld weergeven als we gebruikt hebben bij het invoeren van gegevens in het betaalkaartrecord. Het record wordt opgezocht door het commando FIND en de geheugenvariabele OPZOEK. FIND &OPZOEK betekent het record opzoeken, dat overeenkomt met het kaartnummer opgeslagen in opzoek. De ampersand maakt de computer duidelijk, dat OPZOEK een geheugenvariabele is.

2 Het tweede deel is voor het veranderen van een stortingsrecord. De computer vraagt eerst om de datum waarom het gaat, toont dan de stortingsrecords voor die datum en vraagt vervolgens het nummer van het stortingsrecord dat veranderd moet worden. GOTO OPZOEK zet de database op het record met het recordnummer dat in OPZOEK is opgeslagen. Het veranderingsschermbeeld is weer hetzelfde als dat wat gebruikt is bij het invoeren van gegevens.

MENUKEUZE #4

Via de vierde menuregel kunt u op de betaalkaartenlijst nagaan of u een bepaalde betaalkaart al dan niet hebt uitgeschreven. Figuur 12-8 is een eenvoudige procedure waarmee u de database even kunt doorkijken.

```
USE B:BTLKRTL
ACCEPT 'GEEF MAAND EN JAAR (mm/jj)' TO TIJD
DISPLAY FOR $(DATUM, 4, 2)+$(DATUM, 7, 2)=$(TIJD, 1,2)+$(TIJD, 4, 2)
USE B:BTLKRTL INDEX B:NUMMER
WAIT
RETURN
```

Figuur 12-8 Een procedure om de betaalkaartenlijst door te kijken

Deze procedure laat alle records zien van een bepaalde maand. In dit geval gebruiken we de database zonder indexfile, omdat de indexfile stortingsrecords en betaalkaartrecords scheidt en we ze gewoonlijk samen willen zien op tijdsvolgorde.

De procedure laat (in de volgorde van de recordnummers) de records zien van de aangegeven maand in het aangegeven jaar. De records worden met 15 tegelijk op het scherm gezet (zo werkt DISPLAY). Nadat het vijftiende record op het scherm is gezet, zal het woord WAITING verschijnen. Als u dan op een willekeurige toets drukt, behalve de HOOFDLETTER- of CONTROL toets, zullen de volgende vijftien records op het scherm gezet worden. In het veld DATUM geven het vierde en het vijfde teken de maand aan, terwijl het zevende en het achtste teken het jaar aangeven. In de geheugenvariabele TIJD geven het eerste en het tweede teken de maand aan en het vierde en het vijfde teken het jaar.

De DISPLAY instructie vergelijkt het vierde, vijfde, zevende en achtste teken van DATUM met het eerste, tweede, vierde en vijfde teken van TIJD. Wanneer deze twee stellen tekens met elkaar overeenkomen, zal het record op het scherm worden gezet.

Wanneer de procedure is afgelopen, wordt de file met index weer voor gebruik gekozen.

Het onderwerp van deze menuregel zou een goede kandidaat zijn voor een tweede menu, hoewel we dat hier niet zullen opnemen. Het tweede menu zou een aantal keuzemogelijkheden kunnen bevatten voor bijvoorbeeld het soort RAPPORTEN dat we gemaakt hebben in het voorbeeld van de slijterij in Hoofdstuk 2.

MENUKEUZE #5

Menuregel 5 – een lijst maken van aftrekbare betalingen – geeft u de mogelijkheid een RAPPORTAGEFORMULIER te gebruiken binnen een commandofile. Figuur 12-8A toont de procedure voor het maken van een lijst van aftrekbare betalingen.

```
ACCEPT 'WILT U EEN AFDRUK (Y/N)' TO VRAAG
IF VRAAG = 'Y'
   REPORT FORM B:AFTREK TO PRINT FOR AFTREKBR
ENDIF
IF VRAAG = 'N'
   REPORT FORM B:AFTREK FOR AFTREKBR
ENDIF
RETURN
```

Figuur 12-8A

Het rapport dat dit oplevert, is te zien in Figuur 12-8B. Rapporten zijn een handige manier om gegevens weer te geven.

Page No. 00001
02/01/82

		Aftrekbare kosten	
Kaartnummer	**Begunstigde**	**Datum**	**Bedrag**
208	ST PIJN ZIEKENHUIS	22/06/81	98.50
227	QP COMPUTERS	14/08/81	4011.04
234	DR TAND ARTS	21/08/81	91.00
267	QP COMPUTERS	24/10/81	486.38
289	COMP-U-ZELF	27/11/81	79.50
323	MICROSOF	23/01/82	1257.16
324	QP ELEKTRONICA	23/01/82	344.50
** TOTAL **			6368.08

Figuur 12-8B

MENUKEUZE #6

Menuregel 6 stelt het huidige saldo van de girorekening vast. Deze procedure is bijna hetzelfde als B:SALDO, die we in het vorige hoofdstuk ontwikkeld hebben. Figuur 12-9 laat deze variant zien.

```
USE B:BTLKRTL
STORE 0 TO ONTVANG, UITGAAF
DO WHILE.NOT.EOF
   IF STORTING
      STORE BEDRAG + ONTVANG TO ONTVANG
   ENDIF
   IF.NOT.STORTING
      STORE BEDRAG + UITGAAF TO UITGAAF
   ENDIF
SKIP
ENDDO
@ 5, 15 SAY 'TOTAAL STORTINGEN' GET ONTVANG
@ 7, 15 SAY 'TOTAAL BETAALKAARTEN' GET UITGAAF
STORE ONTVANG-UITGAAF TO SALDO
@ 9, 15 SAY 'SALDO ' GET SALDO
CLEAR GETS
WAIT
USE B:BTLKRTL INDEX B:NUMMER
RETURN
```

Figuur 12-9 Procedure om huidig saldo te berekenen

We gebruiken hier het @ commando van dBASE II om het schermbeeld te krijgen van Scherm 12-5. Een tweede verschil met de oorspronkelijke B:SALDO is het gebruik van het RETURN commando in plaats van het CANCEL commando.

```
          TOTAAL STORTINGEN      :    10677.80
          TOTAAL BETAALKAARTEN   :     9450.51
          SALDO                  :     1227.29

WAITING
```

Scherm 12-5

Omdat het minder tijd kost het saldo te berekenen bij gebruikmaking van de niet-geïndexeerde B:BTLKRTL, gebruiken we de database bij deze procedure niet-geïndexeerd. De tijdsbesparing komt voort uit het sequentieel lezen van records in plaats van heen en weer te springen volgens de index. Een geïndexeerde database vereist heen en weer bewegen en de leeskop voor de diskette moet daarbij ook heen en weer bewegen.

MENUKEUZE #7

De zevende menukeuze controleert de girodienst. U moet op een of andere manier in de gaten houden, hoe de giro met uw rekening omgaat. Deze procedure is één manier om die controle uit te oefenen.

Het bepalen van het saldo gaat in vier stappen:

1 Afgeboekte betaalkaarten afstrepen

2 Bijgeboekte stortingen afstrepen

3 Administratiekosten invoeren

4 Het uiteindelijke saldo vaststellen
al afgeboekte betaalkaarten optellen
al bijgeboekte stortingen optellen
uitkomst vergelijken met de afrekening

De procedure om uw rekening te 'controleren' is te zien in Figuur 12-10. Deze procedure is ook een menu, waarmee u de vier stappen afzonderlijk kunt behandelen en kunt doorgaan, totdat u tevreden bent.

```
STORE ' ' TO KEUZE
DO WHILE.NOT.KEUZE=' 5'
ERASE
@ 5, 15 SAY '(1) Afgeboekte betaalkaarten afstrepen'
@ 7, 15 SAY '(2) Bijgeboekte stortingen afstrepen'
@ 9, 15 SAY '(3) Administratiekosten invoeren'
@ 11, 15 SAY '(4) Saldo van giro vergelijken met eigen berekening'
@ 13, 15 SAY '(5) Terug naar hoofdmenu'
@ 20, 15 SAY 'GEEF UW KEUZE' GET KEUZE
READ
   DO CASE
CASE KEUZE='1'
   DO B:SALMENU1
CASE KEUZE='2'
   DO B:SALMENU2
CASE KEUZE='3'
   DO B:SALMENU3
CASE KEUZE='4'
DO B:SALMENU4
ENDCASE
ENDDO
RETURN
```

Figuur 12-10 Procedure om saldo te vergelijken met afrekening

Als u dit menu vergelijkt met het hoofdmenu voor de betaalkaartenlijst, ziet u een paar verschillen. Dit is een voorbeeld van de algemene regel dat u kunt vaak hetzelfde resultaat op meerdere manieren kunt bereiken.

DO WHILE T is vervangen door DO WHILE .NOT. KEUZE='5'. Het invoeren van 5 beëindigt de DO lus en brengt u terug naar het eerste, het hoofdmenu. Het kiezen van een teken, dat geen menukeuze is, zal het menu opnieuw op uw scherm brengen.

In dit voorbeeld krijgt u een nieuw commando te zien, 'DO CASE'. DO CASE lijkt in beginsel op IF maar wordt gebruikt, wanneer bij een aantal dingen om uit te kiezen, de verschillende keuzemogelijkheden elkaar uitsluiten. Slechts één van de mogelijke gevallen wordt uitgevoerd – ook al zijn er meerdere van toepassing. Het eerste geval dat aan de voorwaarde voldoet, wordt uitgevoerd. Bij een reeks IFs is het mogelijk, dat er meer dan één aan de voorwaarde voldoet en uitgevoerd wordt – ook al is dat niet wenselijk.

Deze opbouw van de menuprocedure verdient de voorkeur boven de opbouw die in het voorbeeld van het hoofdmenu is gebruikt. Het geven van een goede naam aan de procedure maakt herkenning van die procedure gemakkelijk, als u die naam in een ander verband zou tegenkomen.

AFGEBOEKTE BETAALKAARTEN AFSTREPEN

De eerste menuregel maakt het 'afstrepen' mogelijk van betaalkaarten die de giro al verwerkt heeft. In de voorbeeldprocedure, afgebeeld in Figuur 12-11, wordt het record van de betaalkaart opgezocht en worden de relevante gegevens op het scherm weergegeven als 'betaalkaart'. Als u dan het record krijgt, dat u wenste, kunt u het 'afstrepen'. Een representatief voorbeeld van het schermbeeld bij deze procedure is te zien in Scherm 12-6.

```
STORE ' ' TO KRTNAF
STORE 0 TO AANTAL, TOTAAL
DO WHILE.NOT.KRTNAF='X'
   ERASE
   STORE ' ' GET OPZOEK
   @ 5, 10 SAY 'GEEF KAARTNUMMER' TO OPZOEK
   READ
   FIND &OPZOEK
   @ 7, 10 SAY 'BEGUNSTIGDE' GET BGNSTGDE
   @ 9, 10 SAY 'HET BEDRAG IS' GET BEDRAG
   @ 7, 47 SAY 'GEDATEERD' GET DATUM
   CLEAR GETS
   @ 9, 47 SAY 'AFSTREPEN? (Y/N)' GET ALGEBOEKT
   READ
   IF CANCEL
      STORE TOTAAL+BEDRAG TO TOTAAL
      STORE AANTAL+1 TO AANTAL
      @ 12, 10 SAY 'AANTAL AFGESTREEPTE KAARTEN' GET AANTAL
      @ 12, 47 SAY 'TOTAAL' GET TOTAAL
   ENDIF
   CLEAR GETS
   @ 15, 10 SAY 'GEEF X ALS ALLES AFGESTREEPT'
   @ 16, 10 SAY 'RETURN OM DOOR TE GAAN' GET KRTNAF
   READ
ENDDO
RETURN
```

Figuur 12-11 Procedure voor het afstrepen van betaalkaarten

GEEF KAARTNUMMER:**1221:**

BEGUNSTIGDE:Gasbedrijf : GEDATEERD: **29/11/81**

HET BEDRAG IS: **34.56:** AFSTREPEN? (Y/N) :Y:

AANTAL AFGEBOEKTE KAARTEN: **34:** TOTAAL: **574.66:**

GEEF X ALS ALLES AFGESTREEPT RETURN OM DOOR TE GAAN: :

Scherm 12-6

Wanneer u klaar bent met het afstrepen van betaalkaarten, zou het aantal afgestreepte kaarten en het totaalbedrag ervan moeten kloppen met de afrekening van de giro. Zo niet, dan moet u uitzoeken, hoe dat komt. De procedure kan zonodig herhaald worden.

BIJGEBOEKTE STORTINGEN AFSTREPEN

Nu u de procedure voor het afstrepen van betaalkaarten klaar hebt, gaat u door met het 'afstrepen' van stortingen. De procedure hiervoor is B:SAL-MENU2 en is afgebeeld in Figuur 12-12.

```
STORE T TO BEZIG
STORE 0 TO NR, STOR
DO WHILE BEZIG
  ERASE
  REMARK ALS U KLAAR BENT – STOP- INVOEREN ALS RECORDNUMMER
  REMARK
  DISPLAY BEDRAG, DATUM FOR STORTING.AND..NOT.ALGEBOEKT
  ACCEPT 'GEEF AAN TE STREPEN RECORDNUMMER' TO F1
    IF F1 = 'STOP'
      STORE F TO BEZIG
    ELSE
      STORE VAL(F1)TO OPZOEK
      GOTO OPZOEK
      STORE NR + 1 TO NR
      STORE DEP + BEDRAG TO DEP
      REPLACE ALGEBOEKT WITH Y
    ENDIF
ENDDO
  ERASE
  @ 10, 10 SAY 'AANTAL AFGEBOEKTE STORTINGEN' GET NR
  @ 15, 10 SAY 'TOTAAL' GET STOR
  WAIT
  RETURN
```

Figuur 12-12. Procedure voor het afstrepen van stortingen

In dit voorbeeld 'zetten we de DO lus af' vanbinnenuit de procedure. De lus werkt, zolang de geheugenvariabele BEZIG waar is. Merkt u op, dat de T en de F niet door scheidingstekens hoeven te zijn omgeven, omdat de computer al 'aanneemt', dat het LOGISCHE waarden zijn. T staat voor true (waar) en F voor false (onwaar).

Merkt u op, dat het recordnummer opgeslagen werd als tekenketen. (ACCEPT is een manier om de computer te vertellen dat de ingevoerde gegevens tekens zijn.) Als we INPUT zouden gebruiken, konden we het woord 'STOP' niet invoeren in de geheugenvariabele F1. Het is, tussen twee haakjes, niet zo handig T of F te gebruiken als namen van geheugenvariabelen. Soms kan de computer hiervan in de war raken. Als we in dit geval F hadden gebruikt in plaats van F1, dan was de instructie STORE F TO BEZIG dubbelzinnig geweest. Welke 'F' – onwaar of de inhoud van F?

Omdat we het recordnummer hebben opgeslagen als tekenketen, moeten we het weer omzetten in een getal om het te kunnen gebruiken in het commando GOTO. Dit gebeurt met het commando STORE VAL(F1) TO OPZOEK. Dit betekent 'sla de waarde van F1 op in de geheugenvariabele OPZOEK'. Zoals u zou verwachten, kunnen getallen worden omgezet in tekens en kunnen getallen die als tekens zijn opgeslagen, weer omgezet worden in getallen.

HET INVOEREN VAN DE ADMINISTRATIEKOSTEN

De derde procedure in dit menu stelt u in staat de administratiekosten van de giro in te voeren in de database. De procedure waarmee u dat doet, is afgebeeld in Figuur 12-13.

```
ERASE
APPEND BLANK
@ 5, 5 SAY 'GEEF ADMINISTRATIEKOSTEN' GET BEDRAG
@ 7, 5 SAY 'GEEF DATUM (dd/mm/jj)' GET DATUM
READ
REPL STORTING WITH N, BGNSTGDE WITH 'ADMINISTRATIEKOSTEN', ALGEBOEKT WITH Y
RETURN
```

Figuur 12-13 Procedure voor het invoeren van administratiekosten

Merkt u op, dat de administratiekosten als betaalkaarten behandeld zijn, om ervoor te zorgen dat de berekeningen kloppen. Administratiekosten en betaalkaarten gaan beide af van het saldo. Ook administratiekosten worden bij het invoeren 'afgestreept' (als al geboekt aangegeven), omdat ze ook bij de afrekening horen.

Over het tegoed van uw rekening krijgt u rente. Daarom zou bij een 'echte' betaalkaartenlijst ook de mogelijkheid moeten bestaan die rente na te rekenen.

Menuregel 4 verschaft u de mogelijkheid uw versie van de rekening te vergelijken met de afrekening van de giro. Een procedure om dit te doen is afgebeeld in Figuur 12-14.

```
ERASE
INPUT 'GEEF BEDRAG VAN GIRO VOOR SALDO' TO GIROSAL
SUM BEDRAG TO AF FOR ALGEBOEKT.AND..NOT.STORTING
SUM BEDRAG TO BIJ FOR ALGEBOEKT.AND.STORTING
STORE BIJ-AF TO MIJNSAL
@ 10, 11 SAY 'MIJN UITKOMST VOOR HET SALDO' GET MIJNSAL
STORE GIROSAL-MIJNSAL TO FOUT
@ 15, 17 SAY 'DE FOUT VAN DE GIRO IS' GET FOUT
CLEAR GETS
WAIT
RETURN
```

Figuur 12-14 Procedure voor vergelijken van het saldo

In dit voorbeeld moeten we de betaalkaarten en stortingen die de giro nog niet verwerkt heeft, buiten beschouwing laten. In een betaalkaartenlijst 'op papier' telt u die meestal wel mee, omdat u steeds het saldo bijhoudt, als er een mutatie plaatsvindt. In de computerlijst houden we geen rekening met onverwerkte mutaties en gaan we, naar we hopen, uit van dezelfde getallen als de giro. Als alle gegevens goed zijn ingevoerd, zou het schermbeeld eruit moeten komen te zien als in Scherm 12-7.

```
GEEF BEDRAG VAN GIRO VOOR SALDO: 1227.29:

MIJN UITKOMST VOOR HET SALDO:1227.29:

DE FOUT VAN DE GIRO IS:              0:

WAITING
```

Scherm 12-7

In dit hoofdstuk hebben we geprobeerd een volledig database systeem te beschrijven voor het bijhouden van een betaalkaartenlijst. Om de grondbeginselen goed te laten uitkomen hebben we enige details weggelaten. Het menusysteem is een handige manier om de computer te gebruiken voor routinewerk, dat dagelijks, wekelijks of in ieder geval met regelmaat moet gebeuren. Het menu kan submenu's bevatten voor ingewikkelder dingen. In het algemeen moet men ervoor zorgen, dat de meestgebruikte menuregels voorkomen in de 'hoogste' menu's en de minstgebruikte in de 'laagste' menu's.

13. MINDER FOUTEN EN MINDER SAAIHEID MET PROCEDURES

Het nut van uw database is erg afhankelijk van de juistheid van de gegevens erin. Gegevens worden meestal door mensen ingevoerd en mensen maken fouten. Eerder in dit boek hebben we al in het kort computerhulpmiddelen besproken voor het sneller en aangenamer invoeren van gegevens. Vaak bestaat de verleiding de nadruk meer te leggen op snelheid en minder op saaiheid. Kennelijk denkt men aan het terugbrengen van de kosten van het invoeren van gegevens. Helaas is die eventuele 'besparing' vaak maar schijn.

Computerhulp is alleen waardevol als meer gegevens ingevoerd kunnen worden, zonder dat tevens meer fouten optreden. De meeste fouten bij gegevensinvoer zijn of typefouten (dit behoeft geen nadere uitleg) of administratieve fouten, zoals wanneer het oog per ongeluk van de ene regel naar de andere overspringt bij het overbrengen van gegevens van papier naar de computer.

Typefouten komen vaak (maar niet altijd) voort uit verveling. 'Sherry' wordt ingevoerd als 'Sharry'. Vaak worden menu's gebruikt om dergelijke spelfouten te vermijden. Het probleem is, dat als u de verkeerde keuze maakt in het menu, u uiteindelijk 'whiskey' invoert in plaats van 'sherry'. Dit is veel erger dan een eenvoudige spelfout. U kunt een controle invoeren voor dit soort fout.

```
GEKOZEN DRANK WAS SHERRY
JUIST (Y/N)?
```

Dat is nou wel aardig, maar dit beetje onzin kost meer tijd (en moeite) dan gewoon meteen 'sherry' intypen.

U wilt dus dat het invoeren van gegevens snel gaat, maar u vindt het nog belangrijker, dat het soepel en nauwkeurig gebeurt. Een aangename werkomgeving zal al veel meer bijdragen aan een snelle en nauwkeurige gegevensinvoer dan allerlei 'slimme' grappen.

Niettemin is veel van het invoerwerk gewoon routine. Het is een goed idee waar mogelijk de invoer te automatiseren om de saaiheid te verminderen en om bescherming te bieden tegen fouten.

De fout waartegen het gemakkelijkst bescherming te bieden valt, is het invoeren van 'onmogelijke gegevens'. Bij een fout van dit soort vallen de ingevoerde gegevens buiten het bereik van toegestane waarden. Als u bijvoorbeeld de gegevens van leerlingen van een basisschool aan het invoeren bent, dan is er maar een bepaald stel lokalen, waarin een leerling kan zitten. De computer kan dan bescherming bieden tegen het invoeren van een 'niet-toegestaan' lokaalnummer. Hij kan u niet beschermen tegen het per ongeluk invoeren van een geldig maar onjuist lokaalnummer.

Hoe maakt u die bescherming tegen het invoeren van 'onmogelijke gegevens'? Het is niet zo heel simpel, maar moeilijk is het ook niet. Stel dat we een proceduretje hebben voor het invoeren in de database van de naam van de leerling, het lokaalnummer en de klas. Om de zaak eenvoudig te houden heeft onze school maar vier klassen en maar vier lokalen – 101, 102, 103 en 104. Onze voorbeeldprocedure zou er zonder foutcontrole uit kunnen zien als Figuur 13-1.

```
APPEND BLANK
ERASE
@ 10, 10 SAY 'GEEF NAAM LEERLING' GET NAAM
@ 12, 10 SAY 'GEEF LOKAAL' GET LOKAAL
@ 12, 40 SAY 'GEEF KLAS' GET KLAS
READ
RETURN
```

Figuur 13-1

Na het toevoegen van foutcontroles zou de procedure eruit kunnen zien als Figuur 13-2.

```
APPEND BLANK
ERASE
@ 10, 10 SAY 'GEEF NAAM LEERLING' GET NAAM
@ 12, 10 SAY 'GEEF LOKAAL' GET LOKAAL
@ 12, 40 SAY 'GEEF KLAS' GET KLAS
READ
IF.NOT.(LOKAAL = '101'.OR.LOKAAL = '102'.OR.LOKAAL = '103'.OR.
LOKAAL = '104')
ERASE
@ 12, 10 SAY 'U HEBT EEN ONJUIST LOKAALNUMMER
  GEGEVEN'
@ 14, 10 SAY 'GEEF AUB JUISTE LOKAALNUMMER' GET LOKAAL
READ
ENDIF
IF.NOT.(KLAS = '1'.OR.KLAS = '2'.OR.KLAS = '3'.OR.KLAS = '4')
ERASE
@ 12, 10 SAY 'U HEBT EEN ONJUISTE KLAS GEGEVEN'
@ 14, 10 SAY 'GEEF AUB JUISTE KLAS' GET KLAS
READ
ENDIF
RETURN
```

Figuur 13-2

Let u op de opbouw van de logische regels. Dit is de juiste manier om de logica op te zetten, wanneer u alleen bepaalde gegevens in een bepaald veld wilt hebben. U had ook kunnen zeggen

```
IF.NOT.LOKAAL$'101, 102, 103, 104'
IF.NOT.KLAS$'1, 2, 3, 4'
```

Daarmee zou u hetzelfde bereikt hebben.

Als u het gevoel hebt, dat er een redelijke kans is, dat het gegeven ook met deze 'tweede kans' nog niet zal kloppen, kunt u een 'DO' lus gebruiken in plaats van de IF.

```
DO WHILE.NOT.(LOKAAL = '101'.OR.LOKAAL = '102'.OR.LOKAAL = '103'.OR. LOKAAL = '104')
ERASE
@ 12, 10 SAY 'U HEBT EEN ONJUIST LOKAALNUMMER GEGEVEN'
@ 14, 10 SAY 'GEEF AUB JUISTE LOKAALNUMMER' GET LOKAAL
READ
ENDDO
DO WHILE.NOT.(KLAS = '1'.OR.KLAS = '2'.OR.KLAS = '3'.OR.KLAS = '4') ERASE
@ 12, 10 SAY 'U HEBT EEN ONJUISTE KLAS GEGEVEN'
@ 14, 10 SAY 'GEEF AUB JUISTE KLAS' GET KLAS
READ
ENDDO
RETURN
```

Wanneer u een 'DO' lus toepast, kunt u niet verder komen voordat u het goed gedaan hebt.

Een heel praktisch voorbeeld van een veld dat bescherming nodig heeft, is het veld DATUM uit de voorbeelden met de betaalkaartenlijst uit het vorige hoofdstuk. In onze voorbeelden hebben we de datum telkens opnieuw ingetypt bij het invoeren van een betaalkaart of storting. Dat is erg vervelend – in het bijzonder wanneer u een heleboel betaalkaarten uitschrijft. Toch wilden we uitgaande van de datum betaalkaarten op het scherm 'zien'.

De datum minder vaak invoeren is één heel voor de hand liggende manier om de kans op fouten te verkleinen. In dit voorbeeld zouden we de datum eenmaal kunnen invoeren, wanneer we de procedure B:GIROMENU kiezen. Dit gebeurt door de datum op te slaan in een geheugenvariabele en dan het REPLACE commando te gebruiken om de datum in het DATUM veld te zetten.

Een mooi alternatief voor deze benadering is het gebruik van de 'datum' die is ingevoerd toen u dBASE uitkoos. Deze datum kan in het DATUM veld worden ingevoerd met de instructie

```
REPLACE DATUM WITH DATE()
```

Toen u de dBASE datum invoerde, werd die opgeslagen op de geheugenplaats DATE(). Dit is dezelfde datum als 'de datum van de laatste verandering' in uw databasestructuur. Houdt u er wel rekening mee, dat de datum van DATE() de Amerikaanse opbouw heeft met de maand voor de dag.

DATE() neemt acht posities in beslag, net als het DATUM veld in ons voorbeeld. Dat u alle acht tekens iedere keer gebruikt, als u het datumveld invoert, is heel belangrijk. We wilden ons daar in het vorige hoofdstuk niet zo druk over maken. In dat hoofdstuk waren andere belangrijke dingen aan de orde. In dit hoofdstuk gaan we er wel uitgebreid aandacht aan besteden.

In de eerste plaats weet u wel dat

2/12/80 hetzelfde is als 02/12/80,

maar de computer weet dat niet. Hij vergelijkt gewoon de tekens positie voor positie.

```
1  2  3  4  5  6  7  8
-------------------------

2  /  1  2  /  8  0
0  2  /  1  2  /  8  0
```

Omdat tekenvelden links-gealigneerd zijn, begint bij tekenvelden als dit het vergelijken bij het meest linkse teken. Zoals u ziet uit een vergelijking positie voor positie, stemmen de twee tekenketens niet overeen. Voor een heleboel gegevens van dit soort hoeft dit 'probleem' voor u geen probleem te zijn. In dit geval wilt u echter misschien de DATUM gebruiken als sleutel voor het laten weergeven van gegevens. In dit geval kan het heel belangrijk zijn, dat de gegevens steeds op dezelfde manier worden ingevoerd.

Als de kans bestaat, dat gegevens niet op de juiste manier worden ingevoerd en de datum belangrijk is voor het 'ordenen' van records, dan is er een aantal manieren, waarop de computer u kan helpen de gegevens goed in te voeren. Eén mogelijkheid is het gebruik van het commando PICTURE van dBASE II. In plaats van dat de instructie eruit ziet als

```
@ 10, 15 SAY 'GEEF DATUM' GET DATUM
```

gebruikt u

```
@ 10, 15 SAY 'GEEF DATUM' GET DATUM PICTURE '99/99/99'
```

Wanneer de computer de procedure uitvoert, zal hij het volgende neerzetten

```
GEEF DATUM:  /  /  :
```

De scheidingstekens '/' staan op een vaste plaats in de weergegeven regel. Wanneer u getallen invoert voor de datum, slaat de computer de /en over. Nog een voordeel is dat de computer enkel cijfers accepteert op de plaatsen ingenomen door de 9s. Daarmee bent u niet van alle problemen af – maar wel van de meeste en het invoeren van gegevens wordt er gemakkelijker door.

Hoe kunnen we zeker weten dat de gegevens op de juiste manier worden ingevoerd – met of zonder het PICTURE commando? Het blijkt dat we dit gemakkelijk kunnen vaststellen met een procedure die de posities van de tekens in het veld gebruikt. We hebben dat al even genoemd in het vorige hoofdstuk. Nu bekijken we het wat meer in detail. Het is niet moeilijk en het is van onschatbare waarde.

Het derde en het zesde teken moeten /en zijn. Door PICTURE te gebruiken kunnen we garanderen dat ze het ook zijn. Maar zelfs zonder PICTURE kunnen we de datum zo samenstellen als het hoort.

```
1  2  3  4  5  6  7

-------------------------------

d  d  /  m  m  /  j  j        gewenst
d  /  m  /  j  j              mogelijke invoer
```

Er zijn een groot aantal mogelijkheden voor 'mogelijke invoer'. Een eenvoudige (maar wel lange) procedure die alle geldige 'mogelijke invoer' in de gewenste vorm brengt, staat in Figuur 13-3.

```
IF.NOT.$(DATUM, 3, 1)='/'          (als het derde teken
  STORE '0'+DATUM TO DATUM         geen / is, zet dan een
ENDIF                              0 vooraan datum die klopt nu, dan
                                   wel de 'maand' is kleiner dan 10.)
IF.NOT.$(DATUM, 6, 1)='/'          (als het zesde teken geen / is)
  STORE $(DATUM, 1, 3)+'0'+$(DATUM, 5, 4)
  TO DATUM
ENDIF
```

Figuur 13-3

Dit simpele proceduretje zal ervoor zorgen dat de datum de vorm heeft die we wensen.

```
dd/mm/jj
```

In deze procedure zit een nieuwe ingrediënt. Dat is:

```
$(DATUM, 1, 3)
```

Dit betekent 'de eerste drie tekens in het veld DATUM'. Het eerste getal geeft de positie aan van het beginteken. Het tweede getal is het aantal tekens. Dit voorbeeld laat ons zien, dat we met losse tekens kunnen werken, als we dat willen, wat vaak best nuttig is.

Laten we als volgend voorbeeld aannemen dat we de inhoud van onze betaalkaartenlijst een index willen geven op datum. Wanneer we dingen op datum rangschikken, dan gebruiken we

```
jj/mm/dd
```

We denken door nooit erg bij na – maar dit is in feite de manier waarop we dingen op datum sorteren. Anderzijds gebruiken we meestal

```
dd/mm/jj
```

als vorm om data in te voeren. Om de computer de dingen zo op datum te laten zetten, als we dat wensen, zouden we de posities binnen het veld moeten gebruiken. De gegevens kunnen in de goede volgorde worden geïndexeerd met

```
INDEX ON $(DATUM, 7, 2) + $(DATUM, 4, 2) + $(DATUM, 1, 2) TO B:DATUM
```

De indexfile B:DATUM geeft u toegang tot de gegevensrecords op volgorde van de datum.

We moesten de datum uitsplitsen, omdat onze manier om data te lezen voor de computer niet zo gemakkelijk is om te gebruiken. De drie data

```
04/12/81
01/12/82
06/11/82
```

zouden worden 'gerangschikt' als

```
01/12/82
04/12/81
06/11/82
```

als we de computer niet anders instrueren – zoals in het voorbeeld hiervoor. Die voorbeeldindexinstructie zou de data op de juiste manier ordenen tot

04/12/81
06/11/82
01/12/82

De computer is altijd precies en letterlijk. Hij kan alleen werken met de exacte inhoud. Wij mensen zijn associatief – dat wil zeggen wij weten dat Bob Beijers, Robert Beijers en R. A. Beijers een en dezelfde persoon zijn. De computer kan dat niet weten. Hij werkt door te vergelijken. Soms raken wij daar gefrustreerd door. Anderzijds betekent dit dat de computer altijd voorspelbaar is. De regels liggen vast en zijn welomschreven. Als we die regels accepteren, kunnen we heel gemakkelijk nuttige hulp van de computer krijgen.

14. SPECIALE RAPPORTEN UIT DE DATABASE

U kunt gebruik maken van procedures om speciale rapporten te maken met elke willekeurige indeling. U kunt procedures toepassen om rapporten te maken, die niet door de Rapportenschrijver van het DBMS geleverd kunnen worden. Het kost misschien wel een beetje meer moeite dan het samenstellen van een rapport met de Rapportenschrijver van het database systeem (het REPORT commando bij dBASE II). Het kan die moeite dubbel en dwars waard zijn, omdat het rapport precies volgens uw wensen wordt samengesteld.

Om het gebruik van procedures bij het samenstellen van eigen rapporten te verduidelijken gaan we het voorbeeld van een typisch rapport doorwerken, waarvoor de standaard rapportenschrijver niet zo geschikt is.

Een basisschool gebruikt een microcomputer met een database management systeem voor het bijhouden van de gegevens van de leerlingen. Aan het begin van het schooljaar en daarna op gezette tijden is het nodig groepslijsten te verstrekken aan de schoolarts, de bibliothecaresse, de administrateur en de leerkracht.

De database van de school heeft onder andere velden die vastleggen hoever de leerling is met lezen en rekenen, het lokaalnummer, de klas, de naam van de leerkracht en of de leerling is blijven zitten. De relevante delen van de databasestructuur zijn afgebeeld in Figuur 14-1.

Veldnaam	Type	Breedte
NAAM	C	30
KLAS	C	3
LOKAAL	C	1
LEERKRACHT	C	15
ZITTENBL	L	1
LEZEN	N	2
REKENEN	N	2
GESLACHT	C	1

Figuur 14-1 Deel databasestructuur school

De school wil groepslijsten afdrukken en uitdelen, die eruit zien als in Figuur 14-2.

Databasisschool

Leerkracht:Wijsman **9 September 1982**
Lokaal: 101
Klas: 6

Groepslijst schooljaar 1982/1984

Naam	**Lezen**	**Rekenen**	
Aardvarken, Anton	21	88	
Andersen, Hans	43	29	
Appel, Wim	87	29	ZITTENBLIJVER
.	.	.	
.	.	.	
Zappa, Frans	34	19	

GROEPSGROOTTE: 28
JONGENS: 14
MEISJES: 14

GEMIDDELDE LEZEN: 43.32
GEMIDDELDE REKENEN: 62.67

Figuur 14-2 Voorbeeld van een speciaal rapport

Dit voorbeeld kan niet worden gemaakt met de standaardrapportenschrijver van dBASE II (tenminste niet in deze indeling). Het grootste deel van het rapport kan natuurlijk wel door REPORT worden verzorgd, maar de speciale vorm van het rijtje

```
LEERKRACHT
LOKAAL
KLAS
GROEPSGROOTTE
JONGENS
MEISJES
GEMIDDELDE LEZEN
GEMIDDELDE REKENEN
```

kan niet zonder meer door het standaardcommando worden geregeld.

```
SET TALK OFF
USE B:SCHOOL
INDEX ON KLAS + LOKAAL + NAAM TO B:LIJST
USE B:SCHOOL INDEX B:LIJST
ACCEPT 'GEEF DE DATUM' TO DATUM
SET PRINT ON
DO WHILE .NOT. EOF
      ?
      ?
      ?
      ?
      ?
      ?'                                DATABASISSCHOOL'
      ?
      ?'                          LEERKRACHT: ',LEERKRACHT,' ', DATUM
      ?'                          LOKAAL:      ',LOKAAL
      ?'                          KLAS:        ',KLAS
      ?
      ?'                           GROEPSLIJST SCHOOLJAAR 1982/1984'
      ?'                          _____________________________________'
      ?'                            NAAM      LEZEN      REKENEN'
      ?'                          _____________________________________'
      ?
      STORE LOKAAL TO GLOKAAL
      STORE KLAS TO GKLAS
      STORE 0 TO JONGENS, MEISJES, XLEZEN, XREKENEN
DO WHILE LOKAAL = GLOKAAL.AND.KLAS = GKLAS.AND..NOT.EOF
   STORE ' 'TO ZIT
      IF ZITTENBL
         STORE 'ZITTENBLIJVER' TO ZIT
      ENDIF
?' ', NAAM, ' ', LEZEN, ' ', REKENEN, ' ', ZIT
IF GESLACHT = 'M'
      STORE JONGENS + 1 TO JONGENS
   else
      STORE MEISJES + 1 TO MEISJES
ENDIF
STORE XLEZEN + LEZEN TO XLEZEN
STORE XREKENEN + REKENEN TO XREKENEN
SKIP
ENDDO
      ?
      ?'                  GROEPSGROOTTE: ',STR(JONGENS + MEISJES, 2)
      ?'                     JONGENS:          ', STR(JONGENS, 2)
      ?'                     MEISJES:          ', STR(MEISJES, 2)
      ?
      ?'    GEMIDDELDE LEZEN:     ',STR(XLEZEN/(JONGENS + MEISJES), 6, 2)
      ?'    ',STR(XREKENEN/(JONGENS + MEISJES), 6, 2)
```

```
   EJECT
ENDDO
SET PRINT OFF
SET TALK ON
CANCEL
```

Deze procedure zal een reeks afgedrukte rapporten opleveren, waarbij elke groepslijst op een afzonderlijke bladzijde staat. Deze lijsten zijn helemaal aangepast bij de wensen en behoeften van de mensen die ze gebruiken. In de volgende alinea's nemen we deze procedure door en beschrijven we elke stap iets uitgebreider.

Daarbij beginnen we met het hoofdbestanddeel van de procedure. Vervolgens gaan we nieuwe elementen toevoegen en zo gaan we door, totdat we de procedure helemaal af hebben. Bij iedere slag gaan we nieuwe (of gewijzigde) instructies toevoegen in vet lettertype. Het allerbelangrijkste bestanddeel is een do-lus die een aaneengesloten lijst van alle leerlingen van de school maakt (dit is niets anders dan een 'procedure-versie' van het commando LIST).

```
SET TALK OF
DO WHILE .NOT. EOF
   DISPLAY
   SKIP
ENDDO
SET TALK ON
CANCEL
```

Figuur 14-3 Procedure-versie van het dBASE II commando LIST

SET TALK OFF/ON

Bijna alle procedures beginnen en eindigen met dit commando. Wanneer u vanaf het toetsenbord werkt, dan is het wenselijk dat de computer iedere keer antwoord geeft als u een commando invoert. Wanneer u procedures gebruikt, hoeft dat niet. U zult de computer het 'praten' moeten beletten, behalve wanneer u dat wilt. In dit geval willen we de inhoud van records zien (DISPLAY), maar we willen niet dat de computer het recordnummer van het commando SKIP echoot.

SKIP

Elke keer dat dit commando gebruikt wordt, zal de database één record verder komen te staan. Als we geen SET TALK OFF hadden gegeven, zou de computer bij elk gebruik van SKIP het recordnummer laten zien.

Als SKIP wordt toegepast bij een database zonder index, komen de records in de volgorde van hun recordnummers. Als een database met index wordt gebruikt, komen de records in hun 'logische' volgorde.

DO WHILE .NOT. EOF

Geeft de computer opdracht de commando's DISPLAY en SKIP te herhalen, totdat het einde van de database bereikt wordt.

In onze volgende stap bij het verklaren van de procedure voor het speciale rapport zullen we:

De computer zeggen, welke databasefile hij moet gebruiken

De database INDEXeren per groep

De computer zeggen de geïndexeerde database te gebruiken

Alleen de gewenste velden op het scherm halen

```
SET TALK OFF
USE B:SCHOOL
INDEX ON KLAS+LOKAAL+NAAM TO B:LIJST
USE B:SCHOOL INDEX B:LIJST
DO WHILE .NOT. EOF
?NAAM, LEZEN, REKENEN
SKIP
ENDDO
SET TALK ON
CANCEL
```

Figuur 14-4 Gewijzigde procedure-versie van het dBASE II commando LIST

Deze procedure zet nu de leerlingen op het scherm in alfabetische volgorde per lokaal en per klas. Het commando DISPLAY is nu vervangen door het ?. Dit voorkomt dat het recordnummer wordt weergegeven en vraagt minder typewerk dan DISPLAY OFF, dat hetzelfde zou bereiken. Alleen de inhoud van de velden NAAM, LEZEN en REKENEN zal op het scherm komen.

Bij onze volgende stap gaan we de commando's erbij doen voor het maken van een eenvoudige afdruk van de groepslijsten. Bij deze stap in de ontwikkeling van deze procedure, zal de afdruk die we krijgen nog erg ruw zijn. Elke lijst is enkel een opsomming van leerling-namen met cijfers voor lezen en rekenen, welke helemaal bovenaan de bladzijde begint. Er is ook geen linkerkantlijn.

```
SET TALK OFF
USE B:SCHOOL
INDEX ON KLAS + LOKAAL + NAAM TO B:LIJST
USE B:SCHOOL INDEX B:LIJST
SET PRINT ON
DO WHILE .NOT. EOF
   STORE LOKAAL TO GLOKAAL
   STORE KLAS TO GKLAS
   DO WHILE LOKAAL = GLOKAAL.AND.KLAS = GKLAS.AND..NOT.EOF
   ?NAAM, LEZEN, REKENEN
   SKIP
ENDDO
EJECT
ENDDO
SET PRINT OFF
SET TALK ON
CANCEL
```

Figuur 14-5 Basisversie van de groepslijstprocedure

SET PRINT ON/OFF

Hiermee zet u de printer aan en uit. U moet doorgaans de printer aanzetten nadat u SET TALK OFF hebt gegeven. Evenzo zet u de printer uit, voordat u SET TALK ON geeft.

STORE LOKAAL TO GLOKAAL
STORE KLAS TO GKLAS

Deze twee commando's maken het u mogelijk de DO LUS te maken, die van elke groep afzonderlijk een lijst maakt. De computer kan met deze instructies automatisch het begin en het einde van een klassegroep vaststellen.

DO WHILE LOKAAL = GLOKAAL.AND.KLAS = GKLAS.AND..NOT.EOF

We hebben hier te maken met een DO LUS binnen een DO LUS. De binnenste DO LUS staat helemaal binnen de buitenste. Zolang alle records aan de volgende voorwaarden voldoen:

Het lokaalnummer is hetzelfde als GLOKAAL

De klas is hetzelfde als GKLAS

Er wordt geen einde van de file gevonden

zal de procedure de naam van elke leerling en het lees- en rekencijfer blijven afdrukken. Let u op de tweevoudige punt tussen AND en NOT. Dit is de juiste manier om de voorwaarde in te voeren.

EJECT

Dit commando zorgt ervoor dat het papier in de printer doorschuift naar de volgende bladzijde.

Deze procedure is nogal eenvoudig. De buitenste lus maakt het mogelijk van de hele database een lijst te maken. Zodra we de buitenste lus binnengaan (DO WHILE .NOT. EOF) zullen het lokaal en de klas van het eerste record opgeslagen worden in de geheugenvariabele GLOKAAL en GKLAS. Dan gaan we de binnenlus binnen. Deze binnenlus zal herhaald worden, totdat de database bij een record is gekomen, waarin LOKAAL en KLAS niet meer gelijk zijn aan de inhoud van de geheugenvariabelen GLOKAAL en GKLAS. Wanneer dit gebeurt, schuift het papier door en gaan we terug naar de buitenlus – slaan we het nieuwe lokaal en de nieuwe klas op in de geheugenvariabelen en gaan door. Als we het merkteken aan het einde van de file ontmoeten, zetten we de printer af en zijn we klaar. Merkt u op dat ook de binnenlus .NOT.EOF in zijn voorwaarde heeft staan.

Nu zijn we zover, dat we de commando's kunnen toevoegen, die de dingen doen, die nodig zijn om

De jongens te tellen
De meisjes te tellen
De inhouden van de velden met lees- en rekencijfers voor elke groep op te tellen

```
SET TALK OFF
USE B:SCHOOL
INDEX ON KLAS + LOKAAL + NAAM TO B:LIJST
USE B:SCHOOL INDEX B:LIJST
SET PRINT ON
DO WHILE .NOT. EOF
      STORE LOKAAL TO GLOKAAL
      STORE KLAS TO GKLAS
      STORE 0 TO JONGENS, MEISJES, XLEZEN, XREKENEN
DO WHILE LOKAAL = GLOKAAL.AND.KLAS = GKLAS.AND..NOT.EOF
   ?' ', NAAM, LEZEN, REKENEN
   IF GESLACHT = 'M'
      STORE JONGENS + 1 TO JONGENS
   ELSE
      STORE MEISJES + 1 TO MEISJES
   ENDIF
   STORE XLEZEN + LEZEN TO XLEZEN
   STORE XREKENEN + REKENEN TO XREKENEN
   SKIP
   ENDDO
   EJECT
ENDDO
SET PRINT OFF
SET TALK ON
CANCEL
```

Figuur 14-6

```
STORE 0 TO JONGENS, MEISJES, XLEZEN, XREKENEN
```

Dit commando maakt vier geheugenvariabelen

```
JONGENS
MEISJES
XLEZEN
XREKENEN
```

en zet de beginwaarden ervan op nul.

```
IF GESLACHT = 'M'
STORE JONGENS + 1 TO JONGENS
ELSE
STORE MEISJES + 1 TO MEISJES
ENDIF
```

In de binnenste lus tellen we één op bij het aantal JONGENS, als de inhoud van GESLACHT M is. Anders tellen we één op bij MEISJES.

```
STORE XLEZEN + LEZEN TO XLEZEN
STORE XREKENEN + REKENEN TO XREKENEN
```

Verder tellen we de inhoud van het veld lezen op bij de geheugenvariabele XLEZEN en de inhoud van het veld rekenen bij de geheugenvariabele XREKENEN.

Nu zijn we zover dat we de procedure moeten gaan opzetten om de speciaal ingedeelde informatie onderaan alle groepslijsten te kunnen afdrukken.

```
GROEPSGROOTTE
JONGENS
MEISJES
GEMIDDELDE LEZEN
GEMIDDELDE REKENEN
```

Dit gaat met de commando's:

```
?
?' GROEPSGROOTTE:', STR(JONGENS + MEISJES, 2)
?' JONGENS: ', STR(JONGENS, 2)
?' MEISJES: ', STR(MEISJES, 2)
?
?' GEMIDDELDE LEZEN:'STR(XLEZEN/(JONGENS + MEISJES), 6, 2
?' GEMIDDELDE REKENEN:'STR(XREKENEN/(JONGENS + MEISJES), 6, 2)
```

Wanneer het vraagteken (?) in zijn eentje gebruikt wordt, zorgt het voor een blanco regel op het scherm en/of de printer. De commandoregel

```
?'        GROEPSGROOTTE:', STR(JONGENS + MEISJES, 2)
```

zal zorgen voor het afdrukken van een regel die eruit ziet als in het voorbeeld. De spaties vooraan binnen de scheidingstekens dienen om voor een linkerkantlijn te zorgen. Zonder deze spaties zou het woord GROEPSGROOTTE helemaal links op het papier beginnen.

```
STR(JONGENS + MEISJES, 2)
```

Dit deel van het commando neemt de som van de geheugenvariabelen JONGENS en MEISJES en drukt het resultaat af als een tekenketen van twee cijfers. In dit geval weten we dat de uitkomst uit niet meer dan twee tekens kan bestaan. Als we eenvoudigweg de commandoregel zouden schrijven als

```
?'        GROEPSGROOTTE:', JONGENS + MEISJES
```

zou de som worden afgedrukt als een veld van tien cijfers. Daarmee zouden 8 spaties komen te staan tussen de tekst 'GROEPSGROOTTE' en de afgedrukte som.

```
?'        GEMIDDELDE LEZEN:', STR(XLEZEN/(JONGENS + MEISJES), 6, 2
```

Deze versie van het commando werkt net zo, behalve dat de ', 2' vertelt, dat u twee decimale posities wilt zien.

We zijn nu zover, dat we de commando's voor het afdrukken van het bladzijde-opschrift kunnen toevoegen en we de bladzijde op de juiste manier kunnen indelen.

```
SET TALK OFF
USE B:SCHOOL
INDEX ON KLAS + LOKAAL + NAAM TO B:LIJST
USE B:SCHOOL INDEX B:LIJST
ACCEPT 'GEEF DE DATUM' TO DATUM
SET PRINT ON
DO WHILE .NOT. EOF
   ?
   ?
   ?
   ?
   ?
   ?' DATABASISSCHOOL'
   ?
   ?' LEERKRACHT:', LEERKRACHT, ' ', DATUM
   ?' LOKAAL: ', LOKAAL
   ?' KLAS: ', KLAS
   ?
   ?' GROEPSLIJST SCHOOLJAAR 1982/1984'
   ?'————————————————————————'
   ?' NAAM LEZEN REKENEN'
   ?'————————————————————————'
   ?
   STORE LOKAAL TO GLOKAAL
   STORE KLAS TO GKLAS
   STORE 0 TO JONGENS, MEISJES, XLEZEN, XREKENEN
DO WHILE LOKAAL = GLOKAAL.AND.KLAS = GKLAS.AND..NOT.EOF
   STORE ' 'TO ZIT
      IF ZITTENBL
         STORE 'ZITTENBLIJVER' TO ZIT
      ENDIF
   ?' ', NAAM, ' ', LEZEN, ' ', REKENEN, ' ', ZIT
```

(vervolg op de volgende bladzijde)

```
        IF GESLACHT = 'M'
        STORE JONGENS + 1 TO JONGENS
      else
        STORE MEISJES + 1 TO MEISJES
    ENDIF
    STORE XLEZEN + LEZEN TO XLEZEN
    STORE XREKENEN + REKENEN TO XREKENEN
    SKIP
    ENDDO
      ?
      ?' GROEPSGROOTTE:', STR(JONGENS + MEISJES, 2)
      ?' JONGENS: ', STR(JONGENS, 2)
      ?' MEISJES: ', STR(MEISJES, 2)
      ?
      ?' GEMIDDELDE LEZEN:'STR(XLEZEN/(JONGENS + MEISJES), 6, 2)
      ?' GEMIDDELDE REKENEN:',STR(XREKENEN/(JONGENS-
        + MEISJES), 6, 2)
  EJECT
  ENDDO
  SET PRINT OFF
  SET TALK ON
  CANCEL
```

Figuur 14-7

ACCEPT 'GEEF DE DATUM' TO DATUM

Dankzij dit commando kunnen we de datum invoeren in welke vorm we maar willen en verschijnt de aansporing

GEEF DE DATUM:

op het scherm. De computer zal wachten totdat u de datum hebt ingevoerd en op RETURN hebt gedrukt. De datum die u invoert, zal opgeslagen worden in de geheugenvariabele DATUM. Omdat dit commando voor de DO LUS komt, zult u de datum maar eenmaal hoeven in te voeren voor de hele reeks lijsten. U ziet, dat het commando komt voordat de printer wordt aangezet.

BLADZIJDE-OPSCHRIFT

Het bladzijde-opschrift wordt afgedrukt met het ? commando. Een ? dat in zijn eentje gebruikt wordt, zal zorgen voor het afdrukken of op het scherm zetten van een blanco regel. De tekst die is omsloten door enkele aanhalingstekens (scheidingstekens) zal worden afgedrukt als in het voorbeeld. (De scheidingstekens zelf worden niet afgedrukt.) De inhoud van geheu-

genvariabelen of gegevensvelden zal bij gebruik van de naam van de variabele of van het veld worden afgedrukt als afgebeeld.

```
?'      ', NAAM, '      ', LEZEN, '      ', REKENEN, '      ', ZIT
```

Hiermee wordt de basisindeling gemaakt voor elk af te drukken record. De spaties worden gebruikt om de inhoud van de kolommen op de juiste plaats op de bladzijde te krijgen. ZIT is een geheugenvariabele die aangeeft, of een leerling al dan niet is blijven zitten vorig jaar. Die variabele moet gemaakt worden en moet de juiste informatie krijgen voorafgaand aan deze commandoregel.

```
STORE ' 'TO ZIT
IF ZITTENBL
   STORE 'ZITTENBLIJVER' TO ZIT
ENDIF
```

Deze commando's laten één mogelijke manier zien om de geheugenvariabele ZIT voor te bereiden op zijn later gebruik. Deze voorbereiding moet voor elk record herhaald worden.

Zoals u ziet, is er niets moeilijks aan het opstellen van een procedure voor het samenstellen van een speciaal rapport. Alles wat u nodig hebt is zorgvuldigheid en methodiek. Iedere stap die moet worden uitgevoerd, moet in de computer worden ingevoerd. Eén heel goede benadering is het pas invoeren van het commando SET PRINT ON nadat u de procedure aan het werk hebt gekregen. U kunt dan uw procedure uitproberen zonder papier te verknoeien.

Deel 5

DEEL VIJF

Nu we deskundige programmeurs zijn geworden (en u maar denken dat programmeren alleen iets voor computermensen was!), laat Deel Vijf zien, hoe onze databasetoepassingen in een praktische bedrijfssituatie hun werk doen.

Ons voorbeeld van de 'Videotheek' kunnen we gebruiken bij de vele dingen die bij het voeren van een bedrijf komen kijken. Ons database management systeem kan allerlei diensten leveren, variërend van verzendlijsten en voorraadoverzichten tot het vastleggen van gespecialiseerde transacties.

DEEL VIJF

Nu we deskundige programmeurs zijn geworden (en u maar denken dat programmeren alleen iets voor computermensen was!), laat Deel Vijf zien, hoe onze databasetoepassingen in een praktische bedrijfssituatie hun werk doen.

Ons voorbeeld van de 'Videotheek' kunnen we gebruiken bij de vele dingen die bij het voeren van een bedrijf komen kijken. Ons database management systeem kan allerlei diensten leveren, variërend van verzendlijsten en voorraadoverzichten tot het vastleggen van gespecialiseerde transacties.

15. DATABASES IN HET BEDRIJF

Het enige doel van de microcomputer en het database management systeem is u te helpen. Eén heel belangrijk gebied waarop ze kunnen helpen, is het voeren van een klein bedrijf. Kleine bedrijven komen er steeds meer.

Een realistisch voorbeeld van dit soort kleinbedrijf is de videotheek. De Videotheek houdt zich bezig met de verkoop en verhuur van dingen die met video te maken hebben – voornamelijk videorecorders, videocassettes en dat soort dingen. Behalve dat deze winkels goederen verkopen en verhuren, onderhouden ze ook vaak 'videoclubs', waarvan de leden korting krijgen op huur en apparatuur.

Er zijn verschillende bedrijfsaspecten waarbij een database management systeem goed van pas kan komen.

SALARISADMINISTRATIE EN BOEKHOUDING

STANDAARD-BELASTINGAANGIFTEN

VOORRAADBEHEER

HUUR SCHAPRUIMTE

VERZENDLIJSTEN

DAGELIJKSE KASSA-OVERZICHTEN

VASTLEGGEN VAN TRANSACTIES

Deze lijst is natuurlijk niet volledig, maar geeft een goed idee van de dingen die vergemakkelijkt kunnen worden door een DBMS. Het DBMS helpt de gebruiker zijn onderneming beter en met minder inspanning te leiden. We bespreken de videotheek, omdat die bijzonder geschikt is om de beginselen van databasegebruik te verduidelijken. Er zijn maar heel weinig speciale vaardigheden nodig.

VIDEOFILMVERHUUR

Het verhuren van voorbespeelde cassettes met films is de belangrijkste activiteit van de videotheek. De winkel koopt of huurt films om te verhuren. Elke cassette vertegenwoordigt een investering en neemt kostbare ruimte in beslag in de rekken. Het is voor de eigenaar belangrijk te weten, welke films goed verhuurd worden en welke niet. Een film die niet goed loopt, moet weggedaan worden, ofwel door hem in de verkoop te doen, ofwel door hem terug te sturen naar de filmeigenaar. Het is ook belangrijk bij te houden, hoe vaak een populaire film al verhuurd is. Cassettes slijten en 'versleten' films hebben ontevreden (en waarschijnlijk voormalige) klanten tot ge-

volg. Een database management systeem kan zulke dingen bijhouden, zonder dat het tijd kost.

Vele videotheken houden er een 'videoclub' op na. Bij die opzet moet een vast lidmaatschapsgeld betaald worden, waarvoor de leden belangrijke kortingen krijgen op de huurprijs en op andere zaken. Dergelijke 'clubs' werken zowel in het voordeel van de winkel, als in dat van de klant. De lijst van clubleden is vaak het uitgangspunt voor het versturen van reklame. Een database management systeem kan het bijhouden van die lijst vereenvoudigen, etiketten met adressen afdrukken, enzovoorts. Verder kan het systeem bijvoorbeeld helpen te signaleren, wanneer een lidmaatschap vernieuwd moet worden.

Er zijn nog allerlei andere soorten van archief bijhouden, dat bij gebrek aan een computer met potlood en papier wordt gedaan. Om de investering waard te zijn moet een computersysteem moet iets extra doen en/of de hoeveelheid benodigd werk verminderen.

Nieuwe artikelen worden bij ontvangst toegevoegd aan de inventaris en opgenomen in het crediteurensysteem. Bij verkoop moet dit worden vastgelegd en het inventarissysteem moet worden bijgewerkt. Een database systeem kan even goed inventarisbeheer als routinematige boekhoudwerkzaamheden als crediteuren aan.

Het 'uitstallen' van artikelen, dat lijkt op het hebben van goederen in consignatie, is een normale zakelijke regeling. De winkel heeft de goederen voor een bepaalde periode (vaak ongeveer 90 dagen) in huis, voordat er betaald moet worden. Betaling is onmiddellijk vereist, indien het artikel voor de aangegeven datum wordt verkocht. Vaak moet er elke maand een vaste 'uitstallingshuur' betaald worden voor niet-verkochte 'uitstallingsartikelen'. Bij dit soort overeenkomsten moet alles zorgvuldig bijgehouden worden – een database management systeem kan hierbij goed helpen.

Wat zou nou een goed beginpunt kunnen worden voor een computersyteem? Omdat databases met potlood en papier voor iedereen vertrouwd en nuttig zijn, moeten we daar maar van uitgaan. We gaan onze database oude stijl heel letterlijk omzetten in een database op de computer. Bedenkt u eerst eens een overzicht van het werk waarom het gaat.

Op een gewone werkdag zouden een aantal dingen kunnen gebeuren.

Nieuwe artikelen komen binnen
Artikelen worden verkocht
Nieuwe clubleden melden zich aan
Cassettes en apparatuur worden verhuurd
Cassettes en apparatuur worden teruggebracht
De kas wordt opgemaakt

Bij al deze dingen moeten dingen op papier worden bijgehouden. Er moet bijvoorbeeld elke dag een 'kassaformulier' worden ingevuld, waarop de zaken van de dag worden aangetekend en waarop al het geld wordt verantwoord. Een dergelijk formulier is afgebeeld in Figuur 15-1.

KASSAFORMULIER

Weekdag Datum

Ingevuld door

GELD AANWEZIG BIJ OPENING

1. Begintotaal (aanvankelijk in kassa-lade aanwezig) _____

ONTVANGSTEN

2. Huur (bespeelde cassettes en apparatuur) _____
3. Lidmaatschap (levenjaaranders) _____
4. Diensten (reparatie en installatie apparatuur) _____
5. Apparatuur en accessoires vrij van BTW _____
6. Waarborgsommen _____
7. Apparatuur en accessoires inclusief BTW _____

8. Totaal ontvangsten *(2+3+4+5+6+7)* _____
9. Totaal contant terugbetaald (kwitanties bijvoegen) _____
10. Ontvangsten min uitbetalingen *(8 min 9)* _____

GELD AANWEZIG IN KASSA BIJ SLUITING

11. Contanten *(15+16)* _____
12. Bankcheques (aantal) _____
13. Girobetaalkaarten (aantal) _____
14. Totaal in kassa *(11+12+13)* _____

17. TOTAAL *(1+10)* _____
18. Totaal in kassa *(14)* _____

Eventueel verschil tussen #17 en #18 Tekort/Teveel _____

TOTAAL KASSA

Min begincontanten voor kassa-lade volgende dag _____

Naar bank _____

	5,00	
	10,00	
	25,00	
	50,00	
	100,00	
	1000,00	

15. Bankbiljetten

	0,05	
	0,10	
	0,25	
	1,00	
	2,50	

16. Munten

Figuur 15-1

In dit formulier is een groot deel van de dagelijkse administratie van de winkel vervat. Nu gaan we de hierbij passende ondersteuning door database management demonstreren. De eerste stap is de VOORBEREIDING.

Het maken van een ontwerp begint met een goed inzicht in wat er gedaan moet worden. Tot nu toe hebben we het gehad over stukken papier, die gebruikt worden bij de bedrijfsvoering van de winkel.

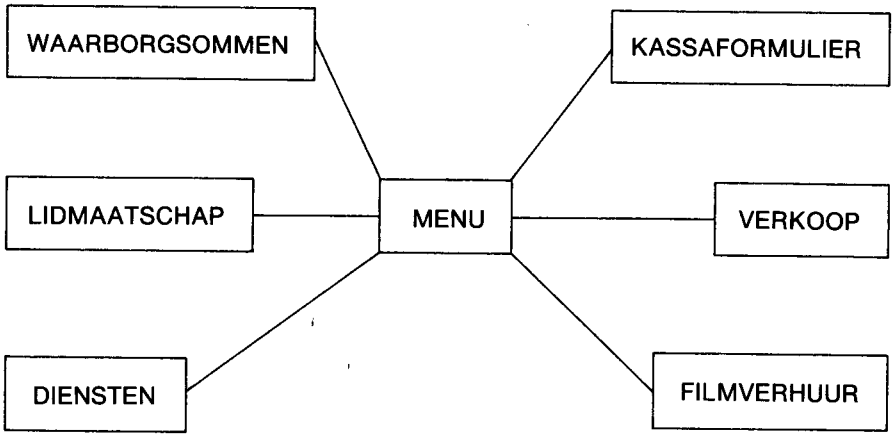
Elk van die stukken papier vervangen door 'elektronisch papier' is een ongecompliceerde en handige manier om een begin te maken. Zo kunt u een systeem in delen opbouwen. U kunt de uitkomsten vergelijken met de administratie op papier, waardoor u de werking van het computersysteem kunt controleren.

In ons voorbeeld begint de winkelbediende met kiezen uit een menu van beschikbare 'elektronische formulieren'. Dit beginmenu is te zien in Scherm 15-1.

```
VOORBEELDMENUSYSTEEM VIDEOTHEEK

1 – FILMVERHUUR

2 – LIDMAATSCHAP

3 – VERKOOP

4 – DIENSTEN

5 – WAARBORGSOMMEN

6 – KASSAFORMULIER

7 – INVENTARIS

8 – STOPPEN
```

Scherm 15-1

Een diagram van hoe deze keuzemogelijkheden bij elkaar passen, staat in Figuur 15-2.



Figuur 15-2

We veranderen eerst het 'papieren formulier' in een 'elektronisch formulier'. Daarvoor hebben we een procedure en een database nodig. Het ontwerp voor de database – B:KASSA – staat in Figuur 15-3.

Veld	Omschrijving	Veldnaam	Type	Breedte	Decimalen
1	Datum	DATUM	C	8	
2	Naam werknemer	NAAM	C	10	
3	In kas begin	BEGKAS	N	6	2
4	Ontvangen huur	HUUR	N	7	2
5	Leden leven	LLEVEN	N	2	
6	Leden jaar	LJAAR	N	2	
7	Leden anders	LANDR	N	2	
8	Ontvangsten	ONLID	N	7	2
9	Ontvangsten	DIENST	N	7	2
10	Verkoop vrij	VVRIJ	N	8	2
11	Verkoop incl.	VIBTW	N	8	2
12	Waarborgsommen	WAARBORG	N	7	2
13	Terugbetaald	TERUG	N	7	2
14	In kas einde	EINDKAS	N	8	2
15	Aantal cheques	CHQAAN	N	3	
16	Aantal girobetaalkaarten	GBKAAN	N	3	
17	Totaal cheques en kaarten	CHQGBK	N	8	2
18	Beginkas volgende dag	BGKASVD	N	6	2
19	Naar bank	BANK	N	8	2

Figuur 15-3. Ontwerp voor kassaformulier

Elk kassaformulier komt overeen met één record van de kassaformulieren-database. Verschillende regels uit het kassaformulier hebben geen overeenkomstige velden in het elektronische formulier, omdat die regels uitkomsten zijn van berekeningen met andere regels uit de database.

Figuur 15-4 laat een eenvoudige procedure zien, waarmee de winkelbediende de velden kan invullen. De regels die beginnen met NOTE worden door de computer genegeerd. Die regels bevatten informatie voor degene die eventueel de procedure moet wijzigen, over hoe de verschillende delen van de procedure werken en waarvoor ze bedoeld zijn.

Bij deze voorbeeldprocedure voert de winkelbediende alle informatie over wat er die dag gedaan is, met de hand in, net als hij zou doen bij een kassaformulier op papier. Het is echter de computer die alle nodige berekeningen maakt.

```
USE B:KASSA
SET TALK OFF
GO BOTTOM
(Nadat u uw database hebt opgezet, moet u tenminste één blanco
record aan de database toevoegen of een paar echte records in de
database hebben, wil dit programma goed werken.)
IF.NOT.DATUM = DATE()
APPEND BLANK
REPLACE DATUM WITH DATE()
ENDIF
ERASE
@ 0, 10 SAY 'KASSAFORMULIER VOOR' GET DATUM
@ 2, 10 SAY 'Naam werknemer' GET NAAM
@ 2, 10 SAY 'In kas bij begin dag' GET BEGKAS
@ 4, 10 SAY 'Inkomsten verhuur' GET HUUR
@ 5, 10 SAY 'Lidmaatschappen leven' GET LLEVEN
@ 6, 10 SAY 'Lidmaatschappen jaar' GET LJAAR
@ 7, 10 SAY 'Lidmaatschappen anders' GET LANDR
@ 8, 10 SAY 'Ontvangsten lidmaatschap' GET ONLID
@ 9, 10 SAY 'Ontvangsten diensten' GET DIENST
@ 10, 10 SAY 'Verkoop vrij van BTW' GET VVRIJ
@ 11, 10 SAY 'Verkoop inclusief BTW' GET VIBTW
@ 12, 10 SAY 'Waarborgsommen' GET WAARBORG
@ 13, 10 SAY 'Terugbetaald' GET TERUG
@ 14, 10 SAY 'In kas bij einde dag' GET EINDKAS
@ 15, 10 SAY 'Aantal cheques' GET CHQAAN
@ 16, 10 SAY 'Aantal girobetaalkaarten' GET GBKAAN
@ 17, 10 SAY 'Totaalbedrag cheques en kaarten' GET CHQGBK
@ 18, 10 SAY 'Beginkas volgende dag' GET BGKASVD
@ 19, 10 SAY 'Naar bank' GET BANK
READ
NOTE nu hebben we alle gegevens om de kassatotalen te berekenen
```

(vervolg op de volgende bladzijde)

```
STORE HUUR + ONLID + DIENST + VVRIJ + VIBTW + WAARBORG TO;
  TOTAALIN
STORE TOTAALIN-TERUG + BEGKAS TO TOTAAL
STORE EINDKAS + CHQGBK TO GELD
ERASE
@ 8, 10 SAY 'IN KASSA BIJ BEGIN DAG' GET BEGKAS
@ 10, 10 SAY 'TOTAAL ONTVANGEN' GET TOTAALIN
@ 12, 10 SAY 'TOTAAL TERUGBETAALD' GET TERUG
@ 14, 10 SAY 'IN KASSA ZOU AANWEZIG MOETEN ZIJN' GET
  TOTAAL
@ 16, 10 SAY 'IN KASSA IS AANWEZIG' GET GELD
DO CASE
  CASE GELD > TOTAAL
  STORE GELD-TOTAAL TO VRSCHL
  @ 18, 10 SAY 'IN KASSA IS TEVEEL' GET VRSCHL
  CASE TOTAAL > GELD
  STORE TOTAAL-GELD TO VRSCHL
  @ 18, 10 SAY 'IN KASSA IS TEKORT' GET VRSCHL
ENDCASE
CLEAR GETS
@ 20, 5 SAY 'DRUK OP EEN WILLEKEURIGE TOETS OM DOOR TE
  GAAN'
WAIT
SET TALK ON
RETURN
```

Figuur 15-4

Het is mogelijk de computer veel meer te laten doen – bijvoorbeeld geld tellen en cheques en betaalkaarten bijhouden net als bij de voorbeelden over de betaalkaartenlijst in Hoofdstuk 11.

We hebben hiervoor in het kort gesproken over de bij dit soort zaak gangbare 'videoclub'-lidmaatschappen. Wanneer een dergelijk lidmaatschap wordt verkocht, vult de winkelbediende een formulier in met:

1 Naam van het lid
2 Adres van het lid
3 Telefoonnummer van het lid
4 Lidmaatschapsnummer in de videoclub
5 Contributie
6 Soort lidmaatschap (leven, jaar, enz.)
7 Datum ingang lidmaatschap

Dit formulier wordt één enkel record in uw databasefile met clubleden, die in dit voorbeeld B:LEDEN heet. Het databaseontwerp voor B:LEDEN staat in Figuur 15-5.

Veld	Omschrijving	Veldnaam	Type	Breedte	Decimalen
1	Datum	DATUM	C	8	
2	Naam lid	NAAMLID	C	30	
3	Straat	STRAAT	C	20	
4	Plaats	PLAATS	C	20	
5	Postcode	PSTCD	C	6	
6	Telefoonnummer	TELEF	C	8	
7	Soort lid	SRTLID	C	1	
8	Contributie	CONTRI	N	6	2
9	Lidnummer	LIDNR	C	8	

Figuur 15-5. Ontwerp voor ledenlijst videoclub

De databaseprocedure die het 'registreren' van een nieuw lid mogelijk maakt, is heel eenvoudig. Hij heet B:LEDEN en staat in Figuur 15-6.

```
USE B:LEDEN
SET TALK OFF
ERASE
GO BOTTOM
STORE VAL(LIDNR) + 1 TO L1
APPEND BLANK
REPLACE LIDNR WITH STR(L1, 8), DATUM WITH DATE()
@ 3, 10 SAY 'Voorbeeld lidmaatschapsformulier op computer'
@ 8, 10 SAY 'LIDMAATSCHAPSNUMMER' GET LIDNR
@ 8, 40 SAY 'DATUM' GET DATUM
CLEAR GETS
@ 10, 10 SAY 'Naam lid' GET NAAMLID
@ 12, 10 SAY 'Straat en huisnummer' GET STRAAT
@ 14, 10 SAY 'Plaats' GET PLAATS
@ 16, 10 SAY 'Postcode' GET PSTCD
@ 18, 10 SAY 'Telefoonnummer' GET TELEF
@ 20, 10 SAY 'Soort lidmaatschap(L-leven J-jaar V-verlenging)' GET;
  SRTLID
READ
DO CASE
  CASE SRTLID = 'L'
    REPLACE CONTRI WITH 100.00
  CASE SRTLID = 'J'
    REPLACE CONTRI WITH 50.00
  CASE SRTLID = 'V'
    REPLACE CONTRI WITH 25.00
ENDCASE
CLEAR GETS
@ 22, 10 SAY 'Contibutie' GET CONTRI
READ
SET TALK ON
RETURN
```

Figuur 15-6

Bij het opstellen van deze procedure komen we iets belangrijks tegen. We kunnen dit lidmaatschapsformulier (als het eenmaal goed werkt) terugkoppelen met het 'kassaformulier'. Een dergelijke koppeling is in dit voorbeeld buitengewoon handig, omdat lidmaatschappen met de hand worden ingevoerd evenals de gegevens voor het kassaformulier. Bij het verkopen van een lidmaatschap kan nu automatisch het kassaformulier bijgewerkt worden.

Dit wordt gedaan met een eenvoudige uitbreiding van de procedure B:LEDEN afgebeeld in Figuur 15-6. Die toevoeging tussen READ en SET TALK ON is te zien in Figuur 15-7.

```
READ (DIT IS DE READ UIT FIGUUR 15-6)
STORE SRTLID TO A1
STORE CONTRI TO A2
USE B:KASSA
GO BOTTOM
IF.NOT.DATUM = DATE()
  APPEND BLANK
  REPLACE DATUM WITH DATE()
ENDIF
DO CASE
  CASE A1 = 'L'
    REPLACE LLEVEN WITH LLEVEN + 1
    REPLACE ONLID WITH ONLID + A2
  CASE A1 = 'J'
    REPLACE LJAAR WITH LJAAR + 1
    REPLACE ONLID WITH ONLID + A2
  CASE A1 = 'V'
    REPLACE LANDR WITH LANDR + 1
    REPLACE ONLID WITH ONLID + A2
ENDCASE
SET TALK ON (OOK DIT KOMT UIT FIGUUR 15-6)
```

Figuur 15-7

In dit voorbeeld van ledenadministratie voegen we alle informatie toe aan de ledendatabase B:LEDEN. Tegelijkertijd kunnen we zonder er zelf ook maar iets voor te hoeven doen het kassaformulier laten bijwerken.

Een belangrijk deel van de omzet bestaat uit het verhuren van films op videocassette. Deze films kunnen het bezit zijn van de winkel, of van derden. Wanneer een film aan een klant wordt verhuurd, vult deze een formulier in en tekent hij het. Het formulier bevat

1 Naam van de klant
2 Adres van de klant
3 Telefoonnummer van de klant
4 Nummer van legitimatie
of
Lidmaatschapsnummer in videoclub
5 Verhuurprijs
6 Eventuele waarborgsom
(clubleden hoeven geen waarborgsom te betalen)
7 Aantal gehuurde films
8 Namen van de gehuurde films
9 Datum begin huur
10 Datum voor terugbrengen

Wanneer de films worden teruggebracht, wordt het formulier 'afgedaan'. De winkel gebruikt dit formulier voor het bijhouden van:

De betaling
Waar de films zijn
Hoe vaak elke film verhuurd is

Op het eerste gezicht lijkt de voor de hand liggende manier om het verhuren op uw computer te behandelen het maken van één record voor elke verhuurhandeling. Het enige probleem bij deze benadering is punt 8. Hoe kunt u een beslissing nemen over de hoeveelheid ruimte die nodig is voor een veld met de (misschien vele) namen van de films? Als we dit op papier op een formulier bijhouden, kunnen we klein schrijven. Met een computer kunt u niet klein schrijven.

De manier om dit probleem op te lossen is het gebruik van meer dan één file voor de verhuurdatabase. De eerste file zal alle benodigde informatie bevatten, behalve de titels van de films. De titels van alle verhuurde films zijn de records van de tweede database. De twee databases zullen gekoppeld worden met een herkenningsnummer voor elke transactie. De transacties moeten allemaal een eigen nummer krijgen. Dat nummer kan één of meer velden in beslag nemen. In dit voorbeeld kan het herkenningsnummer van de transactie worden samengesteld uit het nummer van het legitimatiebewijs van de klant (of zijn lidmaatschapsnummer) en de datum.

De koppeling tussen de twee databasefiles is wat sprekender te zien in Figuur 15-8. De informatie vervat in de twee databases staat in twee kolommen naast elkaar.

	Verhuurdatabase	**Filmtiteldatabase**
1.	Naam klant	
2.	Adres klant	
3.	Telefoonnummer klant	
4.	Verhuurprijs	
5.	Eventuele waarborgsom (niet voor clubleden)	
6.	Aantal gehuurde films	
7.	Aantal huurdagen	
8.	Clublid (j/n)	
9.	Legitimatienummer of Lidmaatschapsnummer	Legitimatienummer of Lidmaatschapsnummer
10.	Datum verhuur	Datum verhuur
11.		Filmtitel
12.		VHS of BETA

Figuur 15-8

Voorlopig is de combinatie van de datum en het nummer van de legitimatie voldoende eenduidig om elke transactie te kenmerken. Elke huurtransactie krijgt een record in de 'verhuurdatabase'. Voor elke film die in de verhuur gaat, komt er ook één record in de 'filmtiteldatabase'. Dit betekent, dat wanneer een klant vier films huurt, vier films zullen worden toegevoegd aan de 'filmtiteldatabase'. Deze titels zijn gekoppeld aan het verhuurrecord via het legitimatienummer en de datum.

We hebben nu de kern van het idee te pakken. Dit voorbeeld vraagt echter veel typewerk van de winkelbediende. Alle informatie over de klant en alle filmtitels moeten worden ingevuld. Gelukkig kunnen we in veel gevallen het typewerk tot een minimum terugbrengen dankzij de informatie die al eerder in de computer is opgeslagen.

Veel huurklanten zullen lid zijn van de videoclub van de winkel. De lidmaatschapsdatabase kan ook worden gekoppeld aan de verhuurdatabase (wanneer het om een lid gaat) via het lidmaatschapsnummer. Een kopie van de informatie uit de lidmaatschapsdatabase kan in de verhuurdatabase worden gezet – waardoor de winkelbediende tijd en moeite spaart (en waarmee tevens de kans op fouten wordt verminderd). Dit is ook een prachtige gelegenheid om niet-leden toe te voegen aan deze database – waardoor het nut ervan als verzendlijst groter wordt. Wanneer een veld gebruikt wordt als koppeling tussen twee databasefiles, moet het veld in beide databasefiles hetzelfde type en dezelfde breedte hebben. Voor de computer is 'Alpha ' niet hetzelfde als 'Alpha'. Hij houdt ook rekening met spaties.

Filmtitels vormen een veld in de inventarisdatabase van verhuurbare films. De voornaamste inventarisinformatie bestaat uit:

Filmtitel
VHS of BETA
Plaats in de rekken
Datum van koop of huur door de winkel
Aankoopprijs/huurprijs
Eigendom of huur
Leverancier

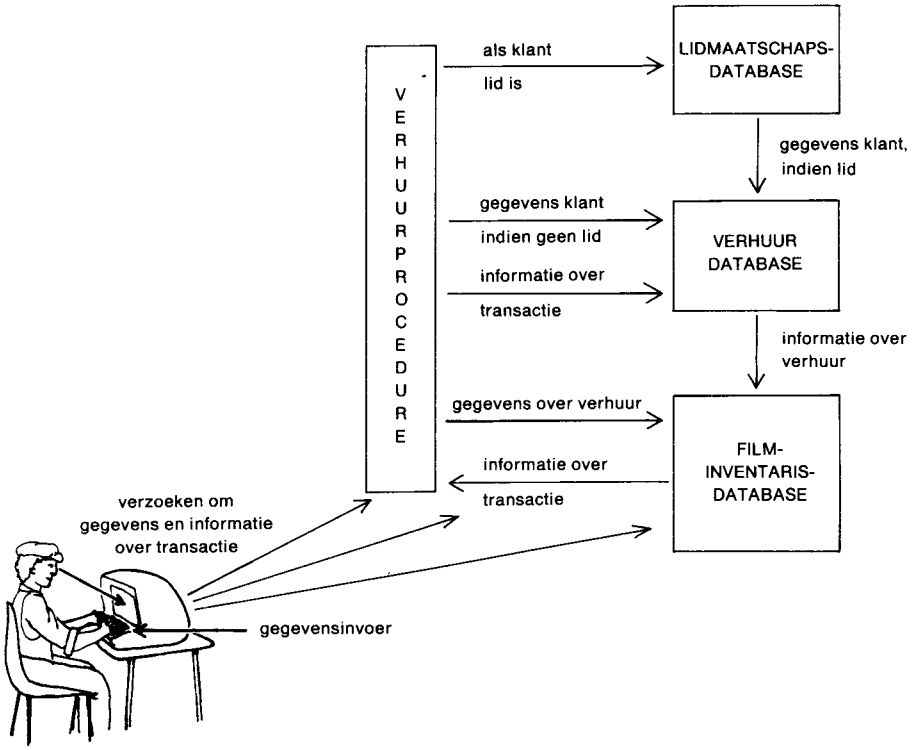
Voor de winkel is de filmtitel wel een lastige manier om met filmrecords om te gaan. Een herkenningsnummer zou de winkelbediende in staat stellen sneller en nauwkeuriger te werken, of dat nou met of zonder computer is. In ons voorbeeld zullen we dus aannemen, dat aan elke te verhuren film een herkenningsnummer van vijf cijfers is toegewezen.

Als we een paar velden toevoegen aan de inventarisdatabase, kunnen we die gebruiken in plaats van de 'filmtiteldatabase'.

Verhuurprijs
Nummer legitimatiebewijs of lidmaatschapsnummer
Verhuurdatum
Aantal malen verhuurd
Nu verhuurd (J/N)

Er zijn een aantal mogelijke combinaties van files, die u voor dit voorbeeld zou kunnen gebruiken. De voor- en nadelen hangen af van wat de toepassing is. In dit voorbeeld besluiten we de inventarisdatabase te gebruiken in plaats van de filmtiteldatabase.

Een diagram van deze verhuurprocedure (voor zover tot nu toe klaar) ziet eruit afgebeeld in Figuur 15-9.

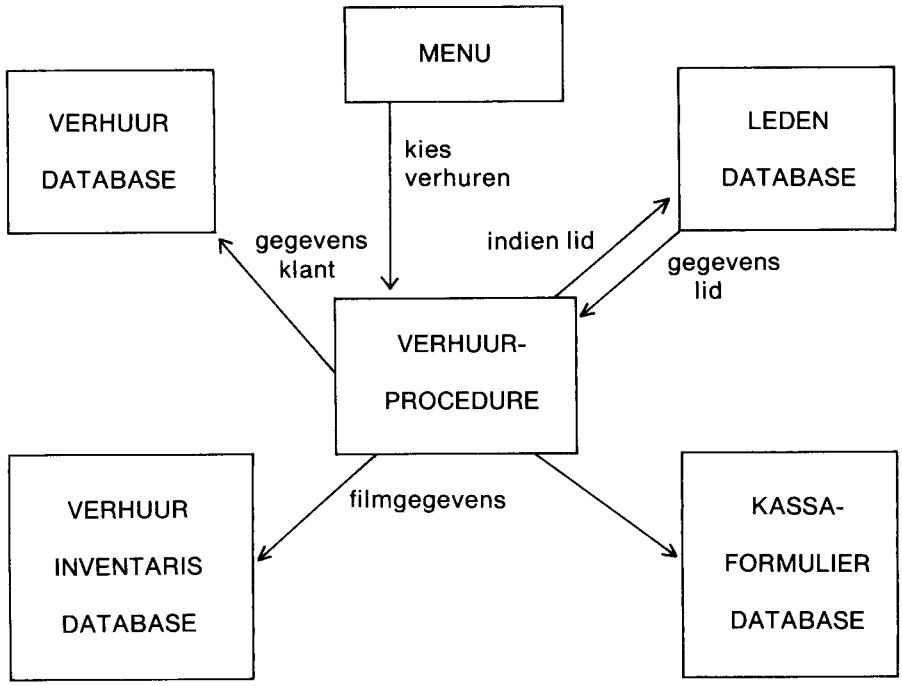


Figuur 15-9

Laten we eens kijken, waar we zijn. Wanneer een klant een film huurt, vult de winkelbediende een 'computerformulier' in. Dit formulier wordt een enkel record in de verhuurdatabase. De inventarisdatabase van verhuurbare films wordt gebruikt om de verhuurde films te identificeren. Het nummer van de legitimatie of het lidmaatschapsnummer en de datum van huur zijn de gemeenschappelijke velden in de twee databasefiles. Deze twee velden dienen om de twee files aan elkaar te koppelen. Iedere keer dat een film wordt verhuurd, wordt één opgeteld bij het veld 'aantal keren verhuurd'. Als de klant lid is van de club, wordt veel van de informatie automatisch uit de lidmaatschapsdatabase gehaald.

Als we nu de onvermijdelijke volgende stap zetten, laten we de computer de totale huurprijs berekenen en toevoegen op de bon. Net als in het vorige voorbeeld kan de waarde van de transactie automatisch worden toegevoegd aan het kassaformulier.

Laten we dit nu eens allemaal bij elkaar zetten. En diagram van dit database systeem staat in Figuur 15-10



Figuur 15-10

Wat hierboven beschreven is, is een goed voorbeeld van één manier om een computersysteem toe te passen bij de administratie van een klein bedrijf. Het is niet de enige manier en het is niet noodzakelijk de beste. Het geeft echter wel inzicht in de manier waarop het inschakelen van een computer in een bedrijf kan worden benaderd.

De volgende stap is het op papier zetten van de databaseontwerpen van alle databasefiles die hierbij gebruikt worden. Het ontwerp van de lidmaatschapsfile werd al beschreven in Figuur 15-5. De twee 'nieuwe' databasefiles staan in de figuren 15-11 en 15-12. Een procedure die het verhuren uitvoert op de hierboven beschreven manier, staat in Figuur 15-13. Toch moeten wij u hier even waarschuwen. Deze procedure is bedoeld om duidelijk te maken, wat er gebeurt. Hij is dan ook in dit voorbeeld wat vereenvoudigd ten opzichte van de procedure die u in een echte zaak zou gebruiken.

Veld	Omschrijving	Veldnaam	Type	Breedte	Decimalen
1	Naam klant	NAAMKL	C	30	
2	Straat en huisnummer	STRAATKL	C	20	
3	Plaats	PLAATSKL	C	20	
4	Postcode	PSTCDKL	C	6	
5	Telefoonnummer	TELEFKL	C	8	
6	Legitimatie	LEGIT	C	8	
7	Huurprijs	HUURPR	N	5	2
8	Waarborgsom	WAARBKL	N	6	2
9	Aantal films	AANFMKL	N	2	
10	Aantal dagen huur	DAGKL	N	2	
11	Clublid (j/n)	LIDKL	L	1	
12	Datum verhuur	DATUMKL	C	8	

Figuur 15-11 Ontwerp voor verhuurdatabase

Veld	Omschrijving	Veldnaam	Type	Breedte	Decimalen
1	Filmtitel	TITEL	C	30	
2	VHS of BETA	VHS	L	1	
3	Plaats	PLANKNR	C	5	
4	Datum binnen-gekregen	ODATUM	C	8	
5	(Huur)prijs	KOSTEN	N	5	2
6	Eigendom of gehuurd	EIGEN	L	1	
7	Leverancier	LEVER	C	20	
8	Verhuurd aan	VNUMMER	C	8	
9	Verhuurprijs	VPRIJS	N	4	2
10	Verhuurdatum	VDATUM	C	8	
11	Aantal malen verhuurd	VTELLING	N	3	
12	Nu verhuurd? j/n)	VERHUURD	L	1	
13	Filmnummer	FILMNR	C	5	

Figuur 15-12. Inventarisdatabase

De procedure die u vindt in Figuur 15-13 is de meest uitgebreide in dit boek. Hij is lang, maar niet moeilijk. De meeste instructies dienen om dingen op het scherm te zetten.

In deze procedure wordt één nieuw commando geïntroduceerd – SELECT. Dit dBASE II commando laat de computer met twee databasefiles tegelijk werken. Tot nu toe hebben we enkel met één database tegelijk gewerkt. Elke keer dat we een database kozen met het commando USE werd het database management systeem op het eerste record van de file gezet. In dit voorbeeld zullen we van de database B:VERHUUR heen en weer gaan naar andere databasefiles. SELECT zal het ons mogelijk maken 'onze plaats bezet te houden' in B:VERHUUR.

Wanneer we twee databasefiles tegelijk gebuiken, heet de ene de PRIMARY file en de andere de SECONDARY. U kunt heen en weer gaan tussen die twee files zonder in een van beide uw plaats te verliezen.

```
USE B:VERHUUR
SET TALK OFF
APPEND BLANK
REPLACE DATUMKL WITH DATE()
ERASE
@ 1, 15 SAY 'VOORBEELD VIDEO-VERHUURFORMULIER'
@ 2, 15 SAY 'IS DE KLANT CLUBLID (Y/N)' GET LIDKL
READ
CLEAR GETS
IF LIDKL
   @ 8, 10 SAY 'GEEF LIDMAATSCHAPSNUMMER' GET LEGIT
   READ
   SELECT SECONDARY          (maakt de computer duidelijk, dat
                              er twee files zullen zijn)
   USE B:LEDEN               (geeft de naam van de tweede file)
   LOCATE FOR LIDNR=LEGIT
   SELECT PRIMARY            (gaat terug naar gewenste record
                              in eerste database)
   Het volgende is de overdracht van gegevens uit het opgezochte
   record in B:LEDEN naar het nieuwe record in B:VERHUUR
   REPLACE NAAMKL WITH NAAMLID, STRAATKL WITH STRAAT,
   PLAATSKL
WITH PLAATS, PSTCDKL WITH PSTCD, TELEFKL WITH TELEF
ENDIF
```

(vervolg op volgende bladzijde)

```
    @ 3, 10 SAY 'NAAM KLANT' GET NAAMKL
    @ 4, 10 SAY 'STRAAT EN HUISNUMMER KLANT' GET STRAATKL
    @ 5, 10 SAY 'PLAATS KLANT' GET PLAATSKL
    @ 6, 10 SAY 'POSTCODE' GET PSTCDKL
    @ 7, 10 SAY 'TELEFOONNUMMER' GET TELEFKL
  IF LIDKL
    CLEAR GETS
  ELSE
    @ 8, 10 SAY 'NUMMER LEGITIMATIEBEWIJS' GET LEGIT
    @ 9, 10 SAY 'GEEF WAARBORGSOM' GET WAARBKL
  ENDIF
    @ 10, 10 SAY 'AANTAL HUURDAGEN' GET DAGKL
    @ 10, 40 SAY 'AANTAL FILMS' GET AANFMKL
    READ
    CLEAR GETS
    SELECT SECONDARY          (kies tweede file of kies nieuwe file)
    USE B:INVENTRS
    STORE AANFMKL TO X
    DO WHILE X>0
      STORE ' ' TO FILMNO
      @ 12, 10 SAY 'GEEF FILMNUMMER' GET FILMNO
      READ
      LOCATE FOR FILMNR=FILMNO
      REPLACE VDATUM WITH DATE(), VTELLING WITH VTEL-
      LING+1,
      REPLACE VERHUURD WITH Y
      REPLACE VNUMMER WITH LEGIT
      SELECT PRIMARY
      REPLACE HUURPR WITH HUURPR+VPRIJS
      STORE X-1 TO X
  ENDDO
  SELECT SECONDARY
  USE B:KASSA
  GO BOTTOM
  IF .NOT. DATUM=DATE()
    APPEND BLANK
    REPLACE DATUM WITH DATE()
  ENDIF
  REPLACE WAARBORG WITH WAARBKL+WAARBORG
  REPLACE HUUR WITH HUURPR+HUUR
  SELECT PRIMARY            (terug naar B:VERHUUR)
  ERASE
```

(vervolg op volgende bladzijde)

```
@ 5, 10 SAY 'NAAM KLANT' GET NAAMKL
@ 7, 10 SAY 'TE BETALEN AAN HUUR VOOR FILMS' GET HUURPR
IF WAARBKL > 0
@ 7, 40 SAY 'VEREISTE WAARBORGSOM' GET WAARBKL
ENDIF
STORE WAARBKL + HUURPR TO TEBETAAL
@ 9, 10 SAY 'TOTAAL TE BETALEN' GET TEBETAAL
CLEAR GETS
SELECT SECONDARY
USE B:INVENTRS
DISPLAY OFF TITEL, VPRIJS FOR VNUMMER = LEGIT.AND.VDATUM-
= DATE()
SELECT PRIMARY
SET TALK ON
RETURN
```

Figuur 15-13

In dit voorbeeld controleert de computer eerst of de klant lid is van de videoclub. Zo ja, dan vraagt hij om het lidmaatschapsnummer, gebruikt de ledenfile en voert de relevante gegevens als naam, adres enzovoorts in in de verhuurfile. Zo nee, dan spoort hij de winkelbediende aan al die gegevens in te voeren. Als dit eenmaal gedaan is, zal hij de winkelbediende aansporen informatie te geven over de transactie. Daarna zal hij de inventarisfile kiezen en de vereiste gegevens invoeren, namelijk

Datum verhuur (VDATUM)
Legitimatie van de klant (nummer legitimatiebewijs of lidmaatschapsnummer)

bij alle gehuurde films. Bovendien zal hij het veld veranderen met het aantal malen dat de cassette verhuurd is, en aangeven dat de film op het moment verhuurd is. Daarna keert hij voor die film terug naar de verhuurdatabase en telt het huurtarief op bij de inhoud van het veld voor de huurprijs (HUURPR). Als dit klaar is, wordt het 'kassaformulier' automatisch bijgewerkt met de informatie over de verhuurinkomsten en de waarborgsom.

Als laatste stap laat de computer de hoofdzaken van de transactie zien, bijvoorbeeld de naam van de klant, het verschuldigde bedrag en de namen van de gehuurde films.

Dit voorbeeld is zeer belangrijk. Er worden vier databasefiles gebruikt. De procedure schermt u (en de winkelbediende) af van wat met de database gebeurt. Na het invullen van de informatie gaat het database management systeem zelf van database naar database, waarbij alle nodige veranderingen en toevoegingen plaatsvinden.

De procedure B:VERHUUR beslaat de hoofdzaken van de filmverhuur-administratie in dit bedrijf. U moet NIET denken, dat deze procedure geschikt is om een echte videotheek mee te leiden. De procedure heeft geen rekening gehouden met allerlei details, die bij het werken in een echte zaak van belang zijn. Sommige dingen zijn weggelaten om de grondbegrippen van het werken met 'formulieren' op het beeldscherm en met meerdere databasefiles beter duidelijk te maken.

EEN PAAR OPMERKINGEN TOT BESLUIT

In dit boek hebben we steeds laten zien, dat database management een gemakkelijke, doeltreffende en natuurlijke manier is om resultaten uit uw computer te krijgen. Bij het verduidelijken van allerlei databasebegrippen hebben we alledaagse voorbeelden gebruikt om de eenvoudige en veelzijdige voorzieningen van een database management systeem te demonstreren.

Hierop terugkijkend is het duidelijk geworden, dat 'database' een begrip is, waarmee iedereen al lang vertrouwd is. De eigen terminologie die gebruikt wordt bij computer database systemen is misschien niet zo vertrouwd, maar moeilijk is die terminologie zeker niet. Het kost niet veel tijd om een kolom gegevens als veld te gaan zien. Ook kost het niet veel moeite de titel van een database als filenaam te zien. Als u zich deze eenvoudige terminologie eenmaal hebt eigen gemaakt, is het gemakkelijk het systeem op zodanig de baas te zijn, dat het voor u werkt.

Dat is, natuurlijk, waar het bij database management systemen voor microcomputers allemaal om draait. Zij zouden het werk moeten doen, terwijl u zich bezighoudt met nadenken. U hoeft geen computerdeskundige te zijn om een computer voor u te laten werken – in dit boek hebt u gezien, dat het echt eenvoudig is de computer in te schakelen. Of een database management systeem nu gericht is op zakelijk of op huishoudelijk gebruik, het zal de beginnende gebruiker een prachtige gelegenheid geven doelmatiger te werken en vertrouwd te raken met computers. Ook de deskundige zal meer bereiken en met voldoening de mogelijkheden van de computer uitbuiten.

Werk met de toepassingen die u het meest opleveren voor uw moeite, terwijl u nog aan het leren bent over uw database management systeem. Denkt u terug aan de inventarislijst van onze slijterij – door de computer te gebruiken, hadden we minder werk te doen, dan wanneer we de inventarisatie zonder computer hadden uitgevoerd. Wanneer u met eenvoudige toepassingen begint, zult u het gemakkelijk vinden verder te gaan met ingewikkelder dingen – de eerste keer dat u ging hardlopen, liep u tenslotte ook geen marathon.

Een veel voorkomende fout is een computer en een database management systeem te kopen, omdat u een ingewikkeld probleem moet oplossen – en dan springt u als eerste begin in het diepe en probeert u dat ingewikkelde probleem op te lossen. Als dit u overkomt, maakt u uw probleem (en uw leven) onnodig ingewikkeld. U moet echt eerst vertrouwd raken met de computer en uw programmatuur. U mag werkelijk niet verwachten dat u het verband tussen uw probleem, de apparatuur en de programmatuur in een keer helemaal kunt leggen (zeker niet als uw probleem ingewikkeld is).

Wij doen u als middel om vertrouwd te raken met uw systeem praktische oefening aan de hand – bedenk 'oefenproblemen' waarbij kleine databases gebruikt worden. De voorbeelden uit dit boek kunnen u in de goede richting helpen bij het opbouwen van uw eigen voorbeelden. DE OPLOSSINGEN VOOR INGEWIKKELDE PROBLEMEN ZIJN OPGEBOUWD UIT VELE EENVOUDIGE STUKKEN. Het is erg nuttig voor u 'oefenproblemen' in elkaar te zetten, die kleine databases gebruiken, omdat u dan uw uitkomsten gemakkelijk kunt controleren en kunt zien of u het 'goede antwoord' krijgt. Die ervaring vormt een stevige grondslag om steeds grotere en grotere databases aan te kunnen. De voorbeelden in dit boek bieden voldoende variatie om u een goede indruk te geven van wat de benadering van een bepaald probleem moet zijn, wanneer u de juiste stukken uit een of meer voorbeelden uitkiest en die vervolgens samenstelt tot een oplossing. Het is weliswaar waardevol de voorbeelden door te werken, maar u zult sneller leren door ze te gebruiken als leidraad bij het uitwerken van een probleem dat van uzelf is.

Als u het 'oefenprobleem' hebt gekozen, waarvoor u de oplossing in elkaar zou willen zetten, moet u dat probleem zorgvuldig bestuderen en een systematisch plan maken voor uw oplossing. Als u uw problemen begrijpt en systematisch te werk gaat bij het plannen en in praktijk brengen van uw oplossing, zult u altijd succes hebben. Deze drie ingrediënten – het probleem begrijpen, planning en systematisch (stap voor stap) de oplossing uitwerken – zijn voorwaarden voor succes met welk computersysteem dan ook.

Succes bij het oplossen van een 'gemakkelijk' beginprobleem zal u de ervaring en het vertrouwen geven om door te gaan met ingewikkelder problemen. U zult daarbij ontdekken, dat die ingewikkelde problemen uiteindelijk niet ingewikkeld zijn. Benader ze stap voor stap, maak u vertrouwd met de taal en u zult er plezier van hebben.

We hopen dat dit boek u goed zal helpen bij het hanteren van dat nuttige en veelzijdige stuk gereedschap – de computer. Database management biedt waarschijnlijk de beste en kortste weg. U zult daarom zeker in de computer een bereidwillige dienaar vinden.

WOORDENLIJST

!

Een symbool dat de computer bij dBASE II vertelt geen onderscheid te maken tussen hoofdletters en kleine letters.

#

Het 'hekje' wordt voor velerlei doeleinden gebruikt. Gewoonlijk dient het als verkorting van 'niet gelijk aan'. Bij dBASE II wordt het ook gebruikt als symbool voor het 'recordnummer'. Voorbeeld: DISPLAY FOR # =3 zet het derde gegevensrecord op het scherm.

$

Een dBASE II operator waarmee u een reeks tekens kunt opsporen binnen een veld of geheugenvariabele. Het symbool wordt vaak deeltekenketenoperator of tekenketenoperator genoemd. Het kan worden opgevat als 'vervat in'. Voorbeeld: met 'Rob'$NAAM kunt u in het veld NAAM de tekenketen ROB opsporen.

*

Wordt bij computersystemen doorgaans gebruikt om vermenigvuldiging aan te geven. Bij dBASE II wordt dit symbool ook gebruikt om databaserecords aan te geven, die ter verwijdering gemerkt zijn.

.AND.

Een logische operator die gebruikt wordt om twee logische uitdrukkingen samen te voegen op zo'n manier, dat de resulterende uitdrukking betrekking heeft op de gemeenschappelijke eigenschappen van de uitdrukkingen. Voorbeeld: KLAS='3'.AND.LOKAAL='122' beperkt de uitdrukking tot derde-klassers, die in lokaal 122 thuishoren.

.NOT.

Een logische operator die gebruikt wordt om het tegenovergestelde aan te roepen van de uitdrukking. Voorbeeld: .NOT.KLAS='3' betekent alle klassen behalve de derde.

.OR.

Een logische operator die wordt gebruikt om twee groepen binnen een logische uitdrukking samen te voegen. De derde en de vierde klas kunnen bijvoorbeeld worden samengevoegd in de logische uitdrukking KLAS='3'.OR.KLAS='4'.

?

Een dBASE II commando voor het laten weergeven van bepaalde dingen.

@
Een dBASE II operator die wordt gebruikt bij de commando's SAY en GET om dingen te laten weergeven op het beeldscherm of op de printer.

ACCEPT TO [naam geheugenvariabele]
Een dBASE II commando dat het invoeren mogelijk maakt van tekenketens in aangegeven geheugenvariabelen, zonder dat scheidingstekens gebruikt hoeven te worden. Wordt meestal gebruikt binnen procedures (commandofiles van dBASE II).

ACHTERGRONDOPSLAG
Randapparatuur voor de opslag van gegevens zoals diskettedrives en cassettedrives.

ADL
Application Development Language – de computertaal die bij dBASE II gebruikt wordt.

APPEND
Het dBASE II commando voor het toevoegen van records aan een database.

APPEND BLANK
Een variant op APPEND, die blanco records toevoegt aan een database. Wordt gewoonlijk gebruikt binnen procedures.

APPEND FROM <filenaam>
Wordt gebruikt om gegevens uit de genoemde file toe te voegen aan de file die in gebruik is.

BACKUP
Een kopie van een diskette of file op een andere diskette, die gemaakt is als bescherming tegen mogelijk onbruikbaar worden van de oorspronkelijke informatie in de toekomst.

BASIC
Een computertaal. Het is een letterwoord dat staat voor Beginners All purpose Symbolic Instruction Code.

BEELDSCHERM
Het videoscherm dat op uw computer is aangesloten.

BESTURINGSSYSTEEM
Programmatuur die uw computerapparatuur toegankelijk maakt voor gebruik.

BOOLESE LOGICA
Een stelsel van computerlogica gebaseerd op het werk van George Boole, die een bepaald soort algebra heeft ontwikkeld.

BOOTEN
Het opstarten van de computer.

BROWSE
Een dBASE II commando dat gebruikt wordt om te kunnen wijzigen. Meerdere records tegelijk worden ter verandering over het hele scherm weergegeven.

BYTE
De hoeveelheid geheugen, die nodig is om een teken als 'A', "#" of '9' op te slaan.

CANCEL
Een commando dat gebruikt wordt om een procedure te beëindigen en de besturing over de computer terug te geven aan het toetsenbord.

CASE
Een variant op de IF instructie. CASE kan enkel worden gebruikt binnen DO CASE/ENDCASE.

CHANGE
Een dBASE II commando dat gebruikt wordt voor het aanbrengen van wijzigingen.

CHR()
Een dBASE II commando waarmee u randapparatuur als de printer en het beeldscherm rechtstreeks kunt besturen.

CLEAR
Dit commando brengt dBASE II in zijn uitgangstoestand terug. Alle databases waarvoor een USE commando gegeven is, worden weer vrijgegeven en het systeem keert terug in dezelfde toestand die het had toen u het programma oorspronkelijk binnenging.

CLEAR GETS
Een commando dat alle wachtende GETs intern verwijdert zonder het scherm te veranderen (zoals het commando ERASE wel zou doen). Hiermee beperkt u het domein van een READ tot enkel die GETs die na het CLEAR GETS commando zijn gegeven.

COBOL
Een computertaal die op grote schaal gebruikt wordt bij zakelijke toepassingen van centrale computers. De eerste op Engels lijkende computertaal. Een letterwoord dat staat voor COmmon Business Oriented Language.

COMMANDO
dBASE II terminologie voor een instructie voor de computer.

COMMANDOFILE
Een verzameling instructies voor de computer, opgeslagen of in bewaring op een diskette voor herhaald gebruik. dBASE II terminologie voor een computerprocedure.

CONTINUE
Een commando dat de database op het volgende record zet, dat voldoet aan de voorwaarden opgegeven in het LOCATE commando.

CONTROLTOETS
Een toets die een derde betekenis kan geven aan alle toetsen op het toetsenbord. Te vergelijken met de hoofdlettertoets die alle toetsen van het toetsenbord een tweede betekenis geeft.

COPY TO <filenaam>
Een dBASE II commando dat een kopie van de database die in gebruik is, in de aangegeven database zet. Wordt gebruikt om een backupkopie van een database te maken. Er bestaan varianten die het u mogelijk maken van slechts bepaalde delen van de in gebruik zijnde database een kopie in de aangegeven database te zetten.

COUNT
Een commando voor de weergave van gegevens, dat het aantal records telt, dat voldoet aan een bepaalde voorwaardelijke uitdrukking.

CP/M
Control Program/Microcomputer. Een veelgebruikt besturingssysteem voor microcomputers. CP/M is een geregistreerd handelsmerk van de Digital Research Corporation.

CPU
Centrale Verwerkingseenheid van het computersysteem. Deze omvat het hoofdgeheugen, de rekeneenheid en bepaalde groepen registers.

CREATE
Het dBASE II commando waarmee u de structuur van een nieuwe databasefile kunt vastleggen.

CRT
Kathodestraalbuis. Wordt gebruikt om het beeldscherm aan te duiden, dat bij computers gebruikt wordt.

CTRL TOETS
Controltoets – wordt gebruikt in combinatie met allerlei toetsen om die een derde betekenis en een derde functie te geven.

DASD
Direct Access Storage Device, bijvoorbeeld een diskette.

DATABASE
Een bewaarplaats voor opgeslagen informatie, die zo geordend is, dat de gegevens gemakkelijk teruggevonden kunnen worden. Wordt gewoonlijk gebruikt in verband met een hoeveelheid gegevens die in een computer is opgeslagen en die voor verschillende toepassingen bruikbaar is. Een alledaags voorbeeld van een database buiten de computer is het telefoonboek.

DBMS
Database management systeem.

DELETE
Een dBASE II commando dat records die weggelaten moeten worden, merkt (zie ook PACK).

DISKETTEDRIVE
Een mechanisch apparaat dat wordt gebruikt om informatie op een diskette te lezen en te schrijven.

DISKETTES
Met magnetisch materiaal bedekte soepele schijven voor het opslaan van gegevens. Het medium waarop een computer tijdelijk of permanent informatie kan opslaan voor later gebruik of uitlezen.

DISPLAY
Een dBASE II commando voor het weergeven van de inhoud van een gegevensrecord.

DISPLAY ALL
Een variant op het commando DISPLAY. Deze variant zal alle records weergeven van de in gebruik zijnde database – het weergeven wordt om de 15 records onderbroken.

DISPLAY FILES ON <drivekenmerk>
Een dBASE II commando dat de filenaam weergeeft, het aantal records en de datum van de laatste verandering van alle databasefiles op de genoemde diskette.

DISPLAY FOR <voorwaarde>
Een variant op het commando DISPLAY. De toevoeging FOR <voorwaarde> wijzigt het DISPLAY commando zo, dat in groepen van 15 alle records zullen worden weergegeven, die aan de voorwaarde voldoen.

DISPLAY MEMORY
Een variant op het commando DISPLAY, die de naam, het type, de afmeting en de inhoud van alle geheugenvariabelen weergeeft.

DISPLAY OFF
Een variant op DISPLAY, die het databaserecord weergeeft zonder recordnummer.

DISPLAY STRUCTURE
Een variant op het commando DISPLAY, die gebruikt wordt om de structuur weer te geven van de op het moment in gebruik zijnde database.

DO <filenaam>
Een dBASE II commando dat de computer opdracht geeft de genoemde procedure (commandofile) uit te voeren.

DO CASE
Een dBASE II commando dat gebruikt wordt als alternatief voor meerdere IF, ENDIF instructies binnen elkaar. Het moet samengaan met een ENDCASE commando. Wordt gewoonlijk gebruikt in procedures.

DO WHILE <voorwaarde>
Een dBASE II commando dat de computer opdracht geeft herhaald de reeks commando's uit te voeren tussen de DO WHILE instructie en de afsluitende ENDDO instructie, zolang aan de voorwaarde is voldaan. Gebruikt in dBASE II commandofiles (procedures).

DOS
Disk Operating System – programmatuur.

EDIT <recordnummer>
Een commando waarmee u de inhoud van gegevensvelden kunt veranderen. Bij dBASE II neemt het commando het hele scherm in bezit, waarna u de inhoud van de gewenste velden kunt veranderen door de cursor naar de gewenste plaats te dirigeren en de nieuwe gegevens in te typen.

EJECT
Een dBASE II commando dat de printer naar een nieuwe bladzijde laat draaien.

ELSE
Biedt in een commandofile een reeks instructies ter uitvoering aan voor het geval de voorwaarde van IF onwaar is.

ENDCASE
Een commando voor commandofiles, dat DO CASE afsluit.

ENDDO
Een commando voor commandofiles, dat DO WHILE afsluit.

ENDIF
Een commando voor commandofiles, dat IF afsluit.

EOF
Einde van file – een dBASE II functie; ook een speciaal teken dat ASCII files afsluit.

ERASE
Een commando dat het beeldscherm wist.

ESCAPE
Een toets die op vele computers gebruikt wordt, waarmee u de uitvoering van een procedure of commando van af het toetsenbord kunt onderbreken. Op vele computers die niet over een ESCAPE toets beschikken, wordt CTRL [gebruikt.

FILE
Een verzameling informatie, zoals een databasefile of een commandofile, die als een herkenbare eenheid op een diskette is opgeslagen.

FILENAAM
Gebruikt om de file voor de computer herkenbaar te maken. Moet bestaan uit acht of minder tekens, moet met een letter beginnen en mag geen spaties bevatten.

FILETYPE
Een toevoeging van drie tekens aan de filenaam die achter een punt op de filenaam volgt. Wordt gebruikt om verschillende typen files met dezelfde naam van elkaar te onderscheiden en verschillende soorten files herkenbaar te maken voor de computer, die zo geprogrammeerd is, dat hij met verschillende soorten files op een verschillende manier omgaat. Een .DBF file bijvoorbeeld wordt door dBASE II herkend als databasefile.

FIND <sleutel>
Een dBASE II commando voor het opzoeken van het record met de sleutel. Kan alleen gebruikt worden bij geïndexeerde files.

FLOPPIES
Floppy disks

FLOPPY DISK
Een opslagmedium dat veel gebruikt wordt bij microcomputers. Informatie wordt opgeslagen op een dunne buigzame mylar schijf. Hetzelfde als een diskette.

FLOPPY DISK SYSTEEM
Een computersysteem dat floppy disks gebruikt.

FORTRAN
Een traditionele computertaal, die voornamelijk gebruikt wordt bij wetenschappelijke toepassingen. Letterwoord voor FORmula TRANslation.

FYSIEKE RECORDS
De feitelijke gegevensrecords.

GEGEVENS
Informatie. Doorgaans alleen in een zekere hoeveelheid nuttig. Het telefoonnummer 08373-25648 is bijvoorbeeld op zichzelf nogal nutteloos. Het bevat wel informatie als het gebruikt wordt samen met een ander gegeven, Jan Klaasen, de naam van de bezitter.

GEHEUGENVARIABELE
Stelt u in staat informatie op te slaan in het hoofdgeheugen van de computer voor tijdelijk gebruik door u. Lijkt in beginsel op het geheugen van een elektronische zakrekenmachine. Alle informatie gaat verloren, wanneer de machine wordt afgezet.

GET
Een dBASE II commando, dat gebruikt wordt samen met het commando @ om de inhoud weer te geven van een veld of geheugenvariabele. Wanneer het gebruikt wordt samen met het commando READ, kan de inhoud van het veld of van de geheugenvariabele worden veranderd door gewoon de nieuwe informatie in te typen.

GETALVELD
Een veld dat getallen bevat, die bedoeld zijn voor gebruik in berekeningen.

GO BOTTOM
Een commando dat ervoor zorgt, dat dBASE II naar het laatste record gaat van de in gebruik zijnde database.

GO TOP
Een commando dat ervoor zorgt dat dBASE II naar het eerste record gaat van de in gebruik zijnde database.

GOTO <recordnummer>
Dit commando wordt gebruikt om de recordwijzer van de database op het genoemde recordnummer te zetten.

I/O
Het accepteren en uitgeven van informatie door de computer met behulp van randapparatuur. Letterwoord voor Input/Output.

IF <voorwaarde>
Een instructie die de computer opdracht geeft een verzameling commando's alleen uit te voeren, als aan een bepaalde voorwaarde voldaan is.

INDEX ON <lijst van velden> TO <filenaam>
Een dBASE II commando dat een indexfile maakt op de genoemde database. De indexfile zorgt ervoor dat de inhoud van de databasefile geordend lijkt in de logische voorwaarde van de inhoud van de genoemde velden.

INPUT TO <naam geheugenvariabele>
Een dBASE II commando waarmee u getallen kunt invoeren in een geheugenvariabele vanaf het toetsenbord. Alleen getallen zullen geaccepteerd worden. Wordt gewoonlijk in procedures gebruikt.

INSERT
Een dBASE II commando dat een nieuw gegevensrecord tussenvoegt middenin een databasefile.

INT()
Een functie die een getal met cijfers achter de komma afbreekt door alles achter de komma (punt) weg te doen. Afkorting voor integer.

JOIN
Een commando dat twee relationele databases samenvoegt.

LEN(naam geheugenvariabele)
Een dBASE II functie die u vertelt, uit hoeveel tekens de genoemde geheugenvariabele van het type tekenketen bestaat.

LIST
Een dBASE II commando voor het achter elkaar weergeven van alle gegevensrecords in een database.

LOCATE FOR <voorwaarde>
Een dBASE II commando voor het opzoeken van een record dat aan de voorwaarde voldoet.

LOGISCHE RECORDS
Records in een indexfile of een daarop gelijkende file, die 'afspiegelingen' zijn van een deel van de feitelijke gegevens. Worden gewoonlijk gebruikt als hulpmiddel bij het gebruik van de eigenlijke gegevens.

LOGISCHE VELDEN
Velden met maar twee mogelijke inhouden. Worden gebruikt wanneer er maar twee elkaar uitsluitende mogelijkheden zijn, zoals 'waar' en 'onwaar', 'ja' en 'nee' of 'man' en 'vrouw'.

LOOP
Een commando dat de uitvoering van een commandofile weer bij het begin van DO WHILE laat vervolgen. Wordt gebruikt als een ontsnappingsmogelijkheid wanneer ongewenste omstandigheden optreden.

MENU
Een computerprocedure voor het weergeven van een aantal keuzemogelijkheden voor verder werken.

MENUSYSTEEM
Een computerprocedure die keuze uit een menu gebruikt om de koers aan te geven die de computer moet volgen.

MICROCOMPUTER
Een kleine computer die voornamelijk is ontworpen voor gebruik door een enkele persoon.

MILLISECONDE
1/1000 deel van een seconde.

MINICOMPUTER
Een kleine computer die in het algemeen is opgezet voor gebruik door een klein aantal mensen tegelijk. Iets groter en met iets meer mogelijkheden dan een microcomputer.

MODIFY COMMAND
Een dBASE II commando waarmee u de inhoud van een procedure (commandofile) kunt invullen en/of wijzigen.

MODIFY STRUCTURE
Een dBASE II commando waarmee u de structuur van een database kunt veranderen. Het veranderen van de structuur zal doorgaans de inhoud van de database vernietigen. Dit commando moet met de nodige voorzichtigheid gebruikt worden.

NOTE <tekst>
Een dBASE II commando (doorgaans voor gebruik in commandofiles), dat ervoor zorgt dat de computer de tekst negeert, die op de regel staat. Wordt gebruikt om de procedure te verklaren voor raadpleging in de toekomst en door anderen.

NUMERIEK VELD
Zie getalveld.

PACK
Een commando dat alle records die gemerkt zijn voor verwijdering, ook daadwerkelijk weghaalt.

PASCAL
Een computertaal die enkele eigenschappen van dBASE II heeft.

PIP
Een CP/M commando waarmee u kopieën van files op de ene diskette op een andere kunt zetten.

PL/1
Een computertaal.

PRIMAIRE SLEUTEL
Een eenduidige manier om een record aan te wijzen, die rechtstreeks bruikbaar is voor het computersysteem. Bij dBASE II is het recordnummer de PRIMAIRE SLEUTEL.

PRINTER
Een randapparaat dat gegevens uit de computer op papier afdrukt.

PROGRAMMA
Een reeks commando's, die de computer als eenheid moet uitvoeren. Een procedure is een programma. Bij dBASE II is een commandofile een programma.

PROGRAMMEUR
Iemand die programma's schrijft.

PROMPT
Hetgeen waarmee de computer aangeeft, dat hij gereed is commando's van het toetsenbord te ontvangen. Ook een verzoek om het invoeren van bepaalde informatie vanaf het toetsenbord.

QUIT
Een commando dat ervoor zorgt dat dBASE II alle op het moment in gebruik zijnde files sluit en de besturing van de computer teruggeeft aan het besturingssysteem.

RAM
Een letterwoord dat staat voor Random Access Memory. Gewoonlijk is dit het hoofdgeheugen van de computer. Random access memory is in feite computergeheugen dat rechtstreeks toegankelijk is en waarin de computer zowel kan schrijven als lezen.

RANDAPPARATUUR
Apparaten die bij de computer worden gebruikt als een printer en diskettedrives.

READ
Een dBASE II commando dat samen met het commando GET wordt gebruikt om op het hele scherm (met de cursor) de inhoud van geheugenvariabelen en gegevensvelden te wijzigen.

RECALL
Een dBASE II commando dat records 'terughaalt', die eerder waren gemerkt voor verwijdering.

RECORD
Een stel bij elkaar horende gegevens. Bij dBASE II is het de informatie die is vervat in een enkele rij van een rechthoekige tabel van rijen en kolommen.

RECORD BLOKKEREN
Een kunstgreep om de mogelijkheid uit te sluiten, dat één enkel gegeven door meerdere gebruikers tegelijk gewijzigd wordt bij systemen waar meer dan één gebruiker tegelijk bezig kan zijn.

RECORDNUMMER
Een herkenningsnummer dat door het database management systeem wordt toegewezen aan elk gegevensrecord. Alle records hebben een verschillend recordnummer (geen tweetal records heeft hetzelfde recordnummer).

RELATIONEEL DATABASE SYSTEEM
Een soort database management systeem dat uitgaat van het gebruik van rechthoekige tabellen van rijen en kolommen. Verschillende databasefiles zijn gekoppeld (hebben een relatie) via de inhoud van een gegevensveld waarvan de inhoud in beide databasefiles hetzelfde is.

RELEASE <namen geheugenvariabelen>
Een dBASE II commando dat de genoemde geheugenvariabelen verwijdert.

RENAME <filenaam1> TO <filenaam2>
Een dBASE II commando dat een file een andere naam geeft. De file kan op dat moment niet in gebruik zijn.

REPLACE <veldnaam> WITH <nieuwe inhoud>
Een dBASE II commando dat de inhoud van het genoemde gegevensveld vervangt door de gewenste nieuwe inhoud. Wordt meestal gebruikt binnen procedures. Kan met vrucht vanaf het toetsenbord gebruikt worden samen met FOR <voorwaarde>.

REPORT
Een dBASE II commando dat op grond van de inhoud van een database een rapport samenstelt met behulp van uw antwoorden op een reeks eenvoudige vragen. Het rapport verschijnt op het beeldscherm. De antwoorden op de vragen worden bij het eerste gebruik van een bepaald rapport vanaf het toetsenbord ingevoerd. De antwoorden worden bewaard in een .FRM file. Het rapport kan in het vervolg dan automatisch samengesteld worden. Vele rapporten kunnen op elk willekeurig moment beschikbaar zijn.

REPORT TO PRINT
Een variant op REPORT, die ervoor zorgt, dat het rapport wordt afgedrukt.

RETURN
Een commando dat een commandofile afsluit en de besturing overgeeft aan de procedure één niveau hoger. De besturing keert terug naar het toetsenbord als er geen hogere procedure is.

RETURN TOETS
Lijkt op de wagenterug-toets op een schrijfmachine. Zou op een computerterminal eigenlijk de INVOER toets moeten heten.

ROM
Een letterwoord voor Read Only Memory, geheugen dat alleen gelezen kan worden.

RUB
Een toets die het teken links van de cursor wist.

SAVE
Een commando dat de op het moment gedefiniëerde geheugenvariabelen wegschrijft naar de achtergrondopslag.

SAY
Een dBASE II commando dat samen met de @ operator wordt gebruikt om informatie weer te geven op het beeldscherm of op de printer.

SCHEIDINGSTEKENS
Een middel om tekenketens voor de computer herkenbaar te maken. De computer kan hierdoor een onderscheid maken tussen het veld NAAM en het woord 'NAAM'.

SEQUENTIËLE TOEGANG
Een methode voor toegang tot gegevensrecords, waarbij de computer alle records achter elkaar onderzoekt – te beginnen bij het eerste record – totdat het gewenste record (of de gewenste records) wordt (worden) gevonden.

SELECT PRIMARY
Een commando dat de PRIMARY database kiest.

SELECT SECONDARY
Een commando dat de SECONDARY database kiest.

SET ECHO ON/OFF
Alle commando's die uit een commandofile komen, worden op het scherm al dan niet geëchood alsof ze vanaf het toetsenbord waren ingevoerd. Gewoonlijk is ECHO uit en moet deze voorziening bij behoefte worden aangezet.

SET PRINT ON/OFF
Stuurt de uitvoer van de computer naar de printer. Gewoonlijk wordt geen uitvoer naar de printer gestuurd.

SET TALK ON/OFF
De uitkomsten van commando's worden gewoonlijk weergegeven op het beeldscherm. Bij gebruik van commandofiles is dit vaak niet gewenst.

SKIP [n]
Een commando dat de database een bepaald aantal records vooruit of achteruit zet.

SORT
Een commando dat de database feitelijk zal herschikken in een of andere gewenste volgorde – gewoonlijk bepaald door de getalwaarde van een bepaald veld of van bepaalde velden of door de alfabetische volgorde van de inhoud van een tekenveld.

STANDAARD
Wanneer de computer een commando krijgt, moet hij iets doen. Tenzij anders is aangegeven, zal hij iets doen op de manier die is voorgeprogrammeerd. Die voorgeprogrammeerde uitvoeringswijze heet de standaard.

STORE
Een commando dat gegevens opslaat in geheugenvariabelen.

STRUCTUUR
De van tevoren vastgelegde opbouw van uw database. Deze wordt vastgelegd door de veldnaam, het veldtype en de veldbreedte.

SUM <veldnaam>
Een commando dat de inhoud van de genoemde velden optelt.

TEKEN
Een symbool van het toetsenbord, dat kan worden afgedrukt, als 'A', '$', '1' of 'a'. (Ook spaties zijn tekens.)

TEKENKETEN
Een aaneengesloten reeks tekens, als 'Willem de Zwijger'.

TEKENVELD
Een veld in een database, dat bedoeld is voor het vastleggen van tekens.

TERMINAL
De middelen waarmee u tegen de computer praat en waarmee de computer tegen u praat. De meest gangbare terminalapparatuur bij microcomputers wordt gevormd door beeldschermen en toetsenborden.

TOETSENBORD
Een apparaat dat lijkt op het toetsenbord van een schrijfmachine, waarmee de gebruiker tegen de computer kan 'praten'.

TYPE
Dit is de functie die het type van gegevens oplevert: voor tekens een 'C' van Character, voor getallen een 'N' van Numeric, voor logische gegevens een 'L' en voor ongedefiniëerde gegevens een 'U' van Undefined.

UPDATE
Een commando dat records uit twee databases samenvoegt.

USE <filenaam>
Een commando dat dBASE II duidelijk maakt, met welke database u wilt werken.

VAL(x)
Maakt het mogelijk de inhoud van een veld of een geheugenvariabele met tekengegevens te gebruiken in berekeningen.

VASTE SCHIJF
Een schijf van onbuigzaam plaatmetaal die bedekt is met een magnetisch film; kan grote hoeveelheden gegevens bevatten.

VASTE-SCHIJFSYSTEEM
Een computersysteem dat vaste schijven gebruikt.

VELD
Bij systemen als dBASE II, bevat een veld een bepaald soort informatie. Het komt overeen met een kolom informatie in een database op papier.

VELDAFMETING
Het aantal posities dat in het veld nodig is om plaats te bieden aan de gegevens.

VELDBESCHRIJVING
De beschrijving van een veld bestaat uit drie delen: de veldnaam, het veldtype en de veldafmeting, inclusief eventuele decimale posities.

VELDBREEDTE
Het aantal posities dat in het veld nodig is om plaats te bieden aan de gegevens die erin moeten komen.

VELDNAAM
Het 'opschrift' van het veld. Bestaat uit tien tekens of minder, moet met een letter beginnen en mag geen spaties bevatten.

VELDTYPE
Het soort gegevens, dat in een veld opgeslagen kan worden. De drie mogelijke veldtypes zijn: tekenveld, getalveld en logisch veld.

VOORWAARDE
Een logische uitdrukking die gebruikt kan worden om een commando als DISPLAY nauwer te omschrijven. DISPLAY FOR NAAM = 'WILLEM DE ZWIJGER' wijzigt het commando DISPLAY dusdanig dat het alleen betrekking heeft op records die voldoen aan de voorwaarde NAAM = 'WILLEM DE ZWIJGER'.

VOORWAARDELIJK VERVANGEN
Een techniek voor het veranderen van de inhoud van een database, waarbij het vervangingscommando wordt beperkt door een voorwaarde. REPLACE LEERKRACHT WITH 'JANSEN' FOR LOKAAL='171' zal de inhoud van het veld LEERKRACHT enkel in die records vervangen door 'JANSEN', die voldoen aan de voorwaarde LOKAAL='171'.

VRAAG
Een verzoek van de gebruiker om informatie van de computer. Gewoonlijk een vraag gesteld vanaf het toetsenbord.

WAIT
Een commando voor commandofiles, dat de verwerking van de commandofile onderbreekt, de aansporing WAITING op het scherm zet en blijft wachten, totdat een enkel teken wordt ingevoerd vanaf het toetsenbord.

WAIT TO <naam geheugenvariabele>
Hetzelfde als WAIT, behalve dat het TO gedeelte ervoor zorgt, dat het in te voeren toetsenbordteken wordt opgeslagen in de aangegeven geheugenvariabele.

WIJZER
Een programmatuurmechanisme dat de computer naar het gewenste databaserecord leidt.

Register

REGISTER