Het converteren van een klinische database van dBase III naar oracle

van Herwijnen, G.J.

Award date:
1988

5321

FACULTEIT DER ELEKTROTECHNIEK
TECHNISCHE UNIVERSITEIT
EINDHOVEN
VAKGROEP MEDISCHE ELEKTROTECHNIEK

HET CONVERTEREN VAN EEN KLINISCHE DATABASE VAN dBASE III NAAR ORACLE

Door G.J. van Herwijnen

Rapport van het afstudeerwerk
uitgevoerd van januari 1988 t/m augustus 1988
in opdracht van prof.dr.ir. J.E.W. Beneken
onder leiding van ir. P.J.M. Cluitmans

SAMENVATTING

Om de uitwisseling van electrofysiologische gegevens tussen specialisten en/of instituten te verbeteren, is het klinische database management systeem (DBMS) EMDABS ontwikkeld. De eerste versie van EMDABS is met behulp van het relationele database systeem dBASE III gerealiseerd. Wanneer grote hoeveelheden gegevens verwerkt moeten worden, blijkt dBASE III te beperkt te zijn. Vandaar dat gezocht werd naar een profesioneler database systeem. De keuze is op Oracle gevallen. Oracle ondersteunt de niet-procedurele taal SQL.
Het implementeren van EMDABS gebeurt met de applicatie generator SQL*FORMS. SQL*FORMS is menugestuurd en maakt gebruik van een screen painter om de schermen te ontwikkelen. Bovendien kunnen triggers gedefinieerd worden, die op bepaalde plaatsen in de form SQL en SQL*FORMS commando's uitvoeren.
EMDABS is menugestuurd en gebruikersvriendelijk. Gegevens worden op hun juistheid getoetst bij het invoeren. De gegevens worden in vier nivo's onderverdeeld, namelijk studie, subject, sessie en tijd. Forms die de studie-, subject- en medicijn-gegevens manipuleren zijn afgerond en worden besproken.

CONVERSION OF A CLINICAL DATABASE FROM dBASE III TO ORACLE

SUMMARY

To facilitate sharing of electrophysiologic data among specialists and/or institutions, the clinical database management system (DBMS) EMDABS was constructed using the relational software system dBASE III. Because of its limited capacity to handle large amounts of data, dBASE III was rejected and a more professional database system was required. The fully relational DBMS Oracle was chosen. Oracle uses the powerful non procedural programming language SQL. EMDABS is implemented using SQL*FORMS, Oracle's application generator. SQL*FORMS is menu driven and contains a screen painter for screen design and triggers, which execute SQL and SQL*FORMS command's at specific points on the forms.
EMDABS is menu driven and users require only minimal instruction. Data integrity is controlled by checks on information as it is entered. EMDABS organizes information in four levels: study, subject, session and time. Forms handling study, subject and drug data are completed and described.

INHOUDSOPGAVE

1 INLEIDING

De vakgroep Medische Elektrotechniek van de Technische Universiteit Eindhoven werkt aan het toepassen van moderne technologische kennis en hulpmiddelen in de gezondheidszorg. Binnen dit kader wordt momenteel gewerkt aan drie projecten, n.l. automatisering in de anesthesie, ultrasone meettechnieken en hulpmiddelen voor gehandicapten.

Als onderdeel van het eerstgenoemde project wordt in samenwerking met de afdeling anesthesie van zowel het academisch ziekenhuis in Nijmegen, als het ziekenhuis van de universiteit van Florida in Gainesville, Florida, onderzoek verricht naar de mogelijkheden om de narcosediepte objectief tijdens operaties te kunnen meten. Zowel een te diepe als een te lichte narcose zijn schadelijk voor de gezondheid van de patiënt. Bovendien zorgt een te diepe narcose voor een langere periode van intensieve en dus kostbare verzorging.

De meeste medicijnen die in de anesthesie gebruikt worden, werken op het centrale zenuwstelsel. Bestudering van dit zenuwstelsel tijdens een operatie kan dan ook wellicht een beeld van de narcosediepte geven. Een manier om dit te realiseren is gebruik maken van zogenaamde "Evoked Potentials" (EP's). Een EP is een aan de schedel geregistreerde responsie van het zenuwstelsel op een stimulus van een perifere zenuw. Deze responsie is zeer zwak zodat hij in het normale elektroencephalogram (EEG) niet zichtbaar is. Door middel van filteren en middelen kan de EP weer uit het EEG gehaald worden. In het huidige onderzoek worden de EP's opgewekt door middel van auditieve stimuli.

Zowel de bij dit onderzoek gemeten fysiologische gegevens als de "real-time events" worden door middel van een speciaal daarvoor ontwikkeld systeem genaamd ERDA (Event Recording and Data Acquisition system) geregistreerd. Onder events worden alle niet automatisch meetbare gebeurtenissen tijdens een operatie verstaan, die relevant zijn voor het verloop van de ingreep. Belangrijk voor een goed verloop van het project is de uitwisseling van deze informatie tussen de verschillende deelnemende instituten. Om de uitwisseling, opslag en evaluatie van gegevens te vergemakkelijken is het database management systeem EMDABS (Electrophysiologic Monitoring DAtaBase System) ontwikkeld (Lit. 12). Aan EMDABS worden de volgende eisen gesteld:

1. Het database management systeem moet geschikt zijn voor verschillende klinische monitoring (bewakings-) apparatuur en computer hardware.
2. Het moet gemakkelijk te gebruiken zijn.
3. Gegevens moeten uitwisselbaar zijn tussen de verschillende specialisten en instituten. Dit betekent

dat een gestandaardiseerde manier van verzamelen en opslaan van gegevens nodig is.

4. Meerdere instituten moeten toegevoegd kunnen worden.

Deze eerste versie van EMDABS is met behulp van het database management systeem dBASE III gerealiseerd. Dit database systeem blijkt, wanneer grote hoeveelheden gegevens verwerkt moeten worden, traag te zijn. Bovendien kan dBASE III maar een beperkt aantal files tegelijkertijd open hebben. Vandaar dat er gezocht is naar een beter database management systeem. De keuze is hierbij gevallen op het professionelere Oracle, een relationeel database systeem dat gebruik maakt van SQL (Structured Query Language).

SQL is een volledige vierde generatie taal en is daardoor niet procedureel. Het biedt naast de mogelijkheden van een normale programmeertaal, zoals gegevensverzamelingen definiëren, opbouwen, wijzigen en raadplegen, ook bijvoorbeeld mogelijkheden op het gebied van de beveiliging en autorisatie.

Mijn afstudeeropdracht bestaat nu uit het converteren van EMDABS van dBASE III naar Oracle. Omdat Oracle een op SQL gebaseerde applicatiegenerator gebruikt, is een (eenvoudige) compilatie van de dBASE III programma's onmogelijk. Vandaar dat een geheel nieuwe implementatie noodzakelijk is.

2 DATABASES

2.1 Inleiding

Een database is een verzameling van gegevens, die op een zodanige manier is georganiseerd, dat aan elke behoefte aan gegevens van de gebruiker voldaan kan worden (Lit. 3).

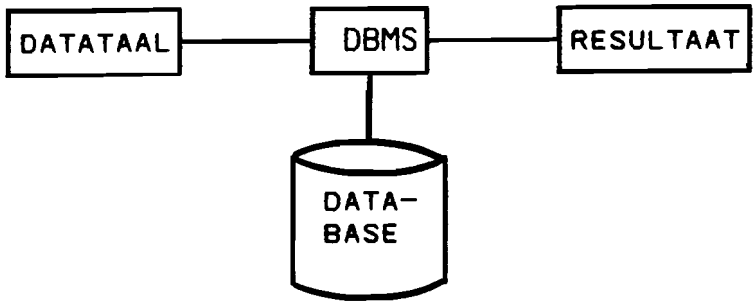
Tussen de fysieke database en de gebruiker bevindt zich het database management systeem (DBMS). Een DBMS is een verzameling van algemeen toepasbare programma's voor het toevoegen, het wijzigen en het opzoeken van gegevens en die voor alle gebruikers de toegang verzorgt. Bovendien zijn bij een DBMS faciliteiten beschikbaar voor data onafhankelijkheid, integriteit en beveiliging. Voorbeelden van een DBMS zijn dBASE III en Oracle.

Voordat gegevens gemanipuleerd kunnen worden, moet eerst de definitie van de gegevensstructuur aan het DBMS kenbaar gemaakt worden. Dit kan met behulp van een zogenaamde datataal. Een datataal bestaat uit twee gedeelten:

1. een definitiegedeelte.
   Hiermee hebben we de mogelijkheid om de gegevensdefinities aan het systeem kenbaar te maken. Deze taal wordt ook wel DDL of data definition language genoemd.

2. een manipulatiegedeelte.
   Hiermee kunnen we gegevens toevoegen, wijzigen en verwijderen. Deze taal wordt ook wel DML of data manipulation language genoemd.

Schematisch ziet de werkwijze er dus als volgt uit:



figuur 2.1. Werkwijze van een DBMS.

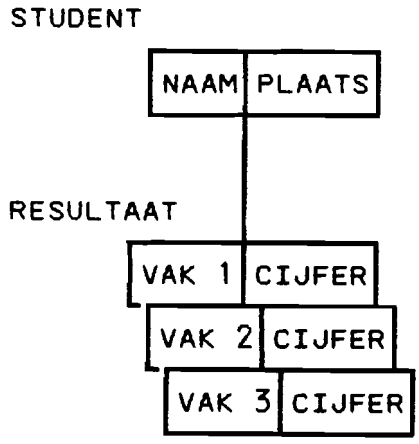
Bij het ontwerpen van een database kunnen we kiezen uit verschillende methoden (benaderingen). Historisch gezien zijn de volgende vier benaderingswijzen relevant:

-hiërarchische benadering.
-netwerkbenadering.
-relationele benadering.
-semantische benadering.

De wijze waarop een DBMS zijn gegevens organiseert is afhankelijk van de gekozen benadering. Het voorbeeld waarmee we deze benaderingen illustreren heeft betrekking op een studieregistratiesysteem, waarbij studenten, studieresultaten en vakken geregistreerd worden. Voor een uitgebreide behandeling van het onderstaande wordt verwezen naar (Lit. 11).

2.2 De hiërarchische benadering

Bij gebruik van de hiërarchische benadering worden de gegevens in een boomstructuur weergegeven. Stel dat bij het bovengenoemde voorbeeld veel toepassingen de gegevens per student zullen gebruiken. Op basis hiervan komen we dan tot een structuur waarbij de gegevens over studenten de ingang tot de overige gegevens vormen. Dit geeft dan de volgende boomstructuur:



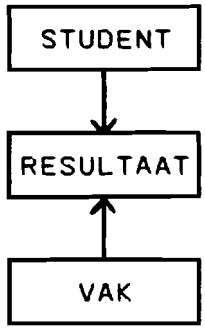
figuur 2.2. Voorbeeld van een boomstructuur bij de hiërarchische benadering.

Het segment student met de velden naam en plaats geeft de wortel van de boomstructuur aan. Het segment resultaat is hieraan ondergeschikt. Tussen deze twee segmenten bestaat een 1:n relatie, dat wil zeggen dat met één student een n-tal resultaten corresponderen. Er onstaan echter problemen wanneer student niet de ingang voor de verwerking is, bijvoorbeeld wanneer we alleen de resultaten per vak willen weten. Het gebruik van de bovenstaande structuur zou ons dan verplichten om de studieresultaten per student te verwerken. Deze beperkte structureringsmogelijkheden resulteren dan ook

vaak in ondoorzichtige algoritmen. Dit heeft geleid tot een betere benadering, namelijk de netwerkbenadering.

2.3 De netwerkbenadering

Bij de hiërarchische benadering kan een segment als ondergeschikte van slechts één ander segment optreden. De betere toepasbaarheid van de netwerkbenadering ligt in het feit dat een segment kan optreden als ondergeschikte van verschillende andere segmenten. Een voorbeeld hiervan is gegeven in figuur 2.3.



figuur 2.3. Voorbeeld van een netwerkstructuur.

Hierbij geldt zowel een 1:n relatie tussen student en resultaat als een 1:n relatie tussen vak en resultaat. Het nadeel van de netwerkbenadering is dat niet alle segmenttypen overeenkomen met de volledige definitie van een bepaald begrip. We kunnen wel zeggen dat de student wordt gekarakteriseerd door de kenmerken naam en adres maar een studieresultaat wordt niet volledig gekarakteriseerd door een cijfer. Daar horen op zijn minst aanduidingen bij van de student en het vak. Het segment resultaat is dus afhankelijk van student en vak. Bij gebruik van de netwerkbenadering vinden we deze gegevens door kettingen af te zoeken. Dit navigeren door de database maakt algoritmen complex. De relationele benadering voorkomt dit bezwaar doordat er naar data onafhankelijkheid gestreefd wordt.

2.4 De relationele benadering

Bij de relationele benadering worden de gegevens alleen in de vorm van tabellen opgeslagen. Codd, de grondlegger van deze benadering, gebruikt het begrip relatie in plaats van tabel. Een tabel bestaat uit een aantal kolommen, met voor ieder attribuut van de te beschrijven objecten één kolom. Elke rij of tupel in de tabel kan opgevat worden als een record, waarbij de velden gevormd worden door de kolomwaarden. De volgorde van de records in een tabel is van geen belang. Elke rij wordt eenduidig geïdentificeerd door de zogenaamde

primaire sleutel (primary key) en is daardoor uniek. Deze sleutel bestaat uit één kenmerk (enkelvoudige sleutel) of uit meerdere (samengevoegde sleutel).

In tegenstelling tot de vorige benaderingen mogen de relaties nu geen impliciete verwijzingen (pointers) bevatten. Verbanden tussen relaties mogen uitsluitend gelegd worden op basis van gemeenschappelijke gegevenswaarden. In ons voorbeeld hebben student en resultaat alleen een onderling verband wanneer ze een gegeven gemeen hebben. In dit geval is het voldoende wanneer de identificatie van student voorkomt in beide relaties. Zodoende krijgen we voor ons voorbeeld de volgende definities, waarbij de sleutels onderstreept zijn:

relatie student (stud_id,naam,plaats)
relatie resultaat (stud_id,vak.id,cijfer)
relatie vak (vak_id,beschrijving)

De basis relationele operatoren zijn:

-selecteren. "select" creëert een deelverzameling van de rijen uit een tabel.
-projecteren. "project" creëert een deelverzameling van de kolommen uit een tabel.
-verbinden. "join" voegt twee tabellen samen.

Zowel dBASE III als Oracle zijn op de relationele benadering gebaseerd. dBASE III is niet volledig relationeel te noemen omdat ten eerste geen direkte manipulatie van tabellen mogelijk is. De dBASE III gebruiker moet namelijk steeds werkgebieden aangeven wanneer hij verschillende tabellen met elkaar wil kombineren. Ten tweede geven de automatisch toegekende recordnummers een fysieke volgorde in de records aan. Zoals al eerder gezegd is de volgorde van de records in een relationele database van geen enkel belang.

2.5 De semantische benadering

Bij de semantische benadering speelt het begrip abstractie een zeer belangrijke rol. Abstractie kan worden opgevat als een beschrijving van de werkelijkheid, waarin enkele details worden benadrukt en tal van details worden genegeerd.

Zoals bij de wiskunde een verzameling gedefinieerd wordt als een samenvoeging van elementen tot een nieuw geheel, zo wordt bij de semantische benadering een type gedefinieerd als een samenvoeging van een zeker aantal eigenschappen tot een nieuw geheel. Niet de elementen, maar de eigenschappen van de elementen spelen dus een belangrijke rol.

Database systemen die op deze benadering gebaseerd zijn, zijn nog in ontwikkeling.

3 DE EERSTE VERSIE VAN EMDABS

Zoals ook al in de inleiding gezegd is, heeft EMDABS tot doel de uitwisseling, opslag en evaluatie van elektrofysiologische gegevens en "real-time events" mogelijk te maken. Voorbeelden van deze gegevens zijn: elektrocardiogram (ECG), arteriele bloeddruk, pulmonale en centraal veneuze druk (PA/CVP), temperatuur, elektroencephalogram (EEG), "evoked potentials" en/of de afgeleide parameters van deze grootheden.

3.1 dBASE III

Zoals al eerder gezegd is, bezit dBASE III een relationele database structuur. De database zelf is een verzameling van data files en index files, waarbij de laatste gebruikt worden om het zoeken te versnellen. Per data file kunnen maximaal 7 index files gebruikt worden. Het nadeel van dBASE III is dat slechts 10 data files tegelijk gebruikt kunnen worden. Wanneer ook andere files meegerekend worden, zoals programma files en index files, kunnen maximaal 15 files tegelijk gebruikt worden (Lit. 2). Omdat EMDABS veel programma files en data files (met bijbehorende index files) gebruikt, is het niet verwonderlijk dat de aangegeven limiet van 15 files overschreden kan worden.

De in hoofdstuk 2 genoemde basis relationele operatoren selecteren, projecteren en verbinden, zijn ook in dBASE III te gebruiken. Het selecteren van rijen gebeurt met het DISPLAY FOR commando. Wanneer we gebruik maken van de eerdergenoemde studenttabel, kunnen we bijvoorbeeld de volgende selectie verrichten:

```
DISPLAY ALL stud_id FOR plaats="EINDHOVEN"
```

Het projecteren van een of meerdere kolommen kan ook met behulp van DISPLAY gerealiseerd worden:

```
DISPLAY ALL stud_id, naam
```

Een gecombineerde selectie en projectie ziet er dan als volgt uit:

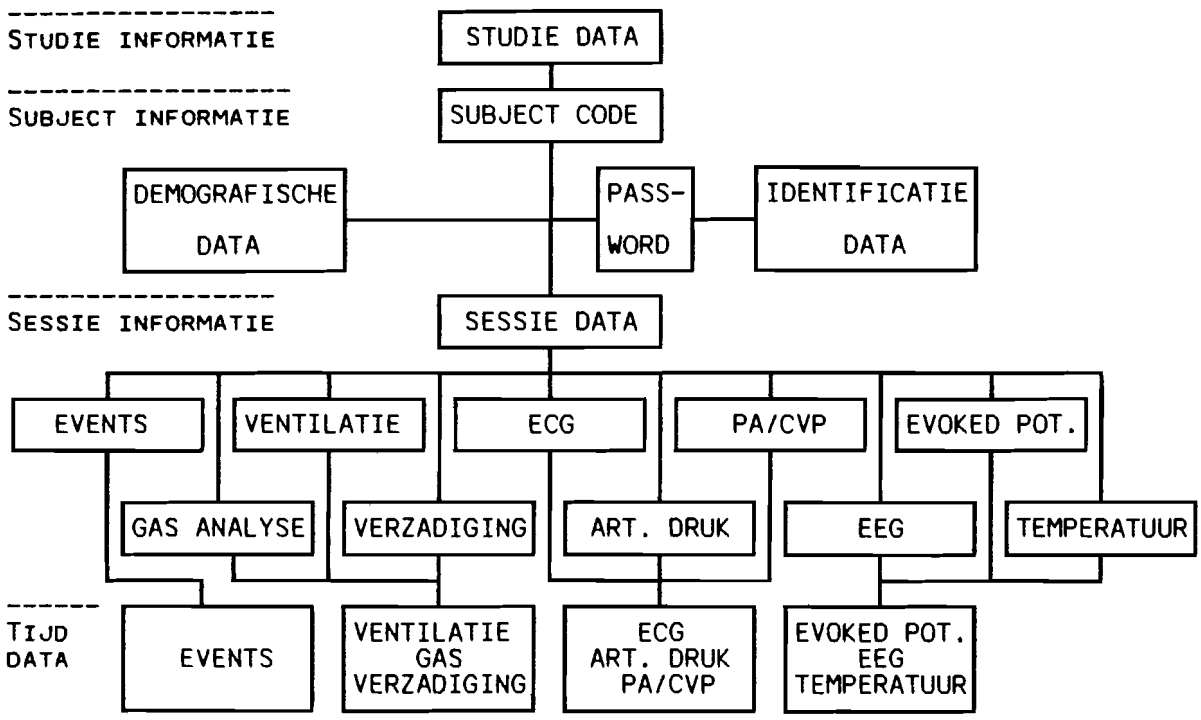
```
DISPLAY ALL stud_id, naam FOR plaats="EINDHOVEN"
```

Het verbinden van twee data files (relaties) vindt plaats door middel van het JOIN WITH commando.

3.2 De structuur van EMDABS

De eerste versie van EMDABS is, vooral vanwege de eenvoudige programmeertaal, met behulp van het relationele DBMS dBASE III gerealiseerd (Lit. 12). EMDABS is menugestuurd en de informatie wordt door de gebruiker in een gestandaardi-

seerde vorm ingevoerd. De integriteit van de gegevens wordt hierbij gecontroleerd. EMDABS organiseert de informatie op vier nivo's, namelijk studie, subject, sessie en tijd (zie figuur 3.1).



figuur 3.1. De structuur van EMDABS.

Een studie kan gezien worden als een onderzoeksproces. Op het studie nivo worden de studie code, de titel van de betreffende studie en de datum ingevoerd. Bovendien wordt ook aan elke studie een code toegekend, die voor elk instituut uniek is. Elke studie kan verschillende subjecten bevatten. Een subject kan zowel betrekking hebben op een patiënt als op een proefdier.

Op het subject nivo wordt een bestaande subject code ingevoerd, of wordt een voor die studie unieke subject code toegewezen. Ook is het mogelijk om op dit nivo zowel demografische informatie (bijvoorbeeld leeftijd, lengte en gewicht) als identificatie informatie (bijvoorbeeld naam en ziekenfondsnummer) toe te voegen.

Het derde nivo behandelt de bij elk subject behorende sessie

informatie. Voor elke sessie worden de plaats, datum en tijd van de sessie, de deelnemende medici, de gebruikte monitoring apparatuur en de sample frekwenties ingevoerd. Voorbeelden van een sessie zijn een operatie en een meetserie tijdens een labaratorium studie.

De bij elke sessie behorende metingen en waarnemingen vormen het tijd nivo. Dit laatste nivo bevat de door de verschillende monitoring apparatuur gegenereerde data en de events die gedurende de sessie plaatsvinden. Deze gegevens worden met behulp van het eerdergenoemde ERDA (Event Recording and Data Acquisition system) verzameld, en, na een eventuele off-line data reduktie, aan EMDABS aangeboden. Omdat niet alle parameters, zoals bijvoorbeeld ECG, EEG en EP's, bij elke studie gemeten worden, worden op het tijd nivo gegevensbestanden geconstrueerd, die alleen die parameters bevatten, die gedurende die sessie zijn geregistreerd.

Studie, subject en sessie gegevens kunnen zowel voor als na de registratie van de tijdgegevens ingevoerd worden.

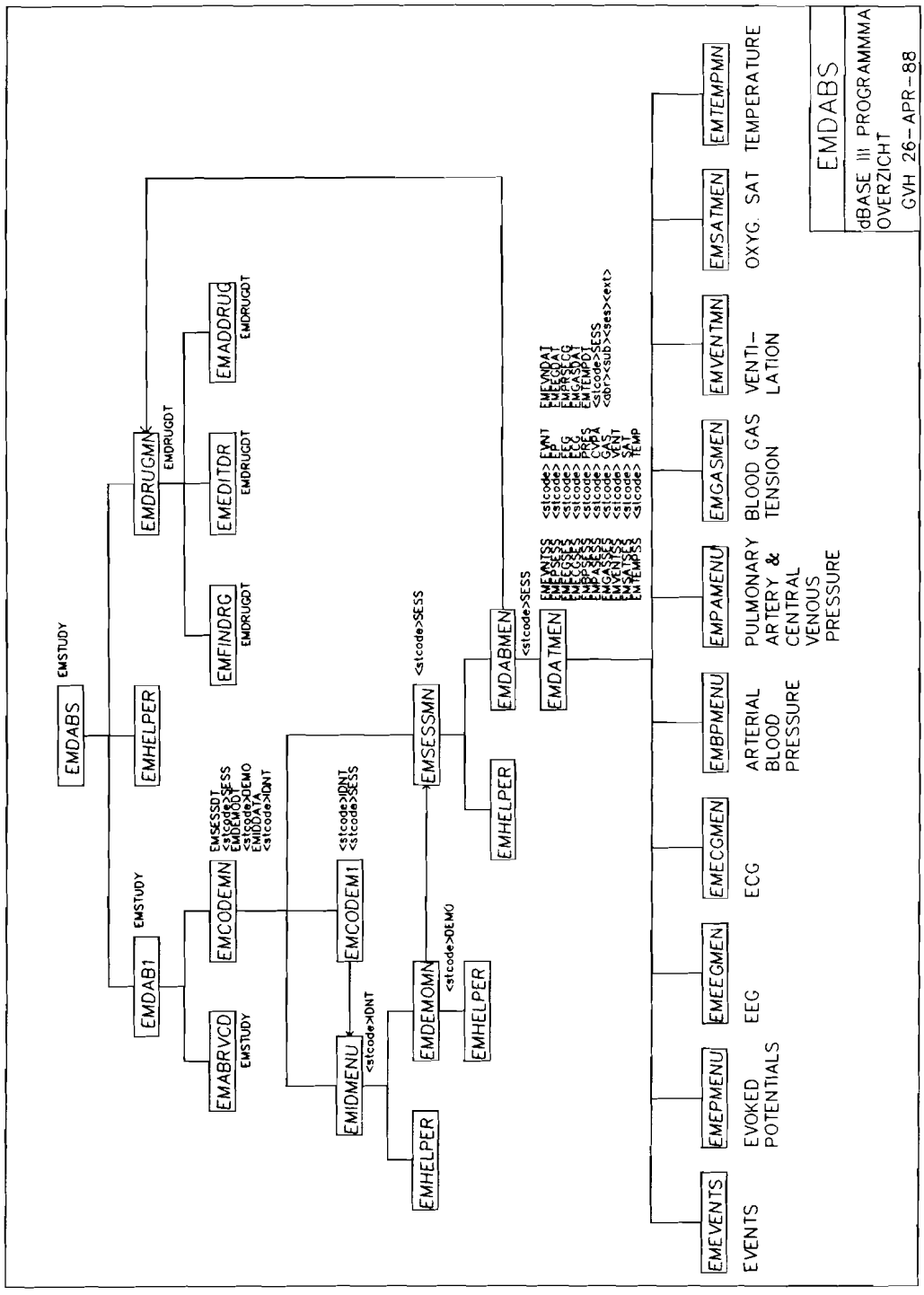
3.3 Analyse van de dBASE III programma's

Jammer genoeg zijn de dBASE III programma's, waaruit deze eerste versie van EMDABS is opgebouwd, slecht of in het geheel niet gedokumenteerd. Om er nu toch achter te komen hoe EMDABS precies werkt ben ik de dBASE III programma's gaan ontleden. Daarbij heb ik het programma overzicht opgesteld dat is weergegeven in figuur 3.2. In de rechthoeken staan de namen van de programma's en daarbij de gebruikte database file(s). De listings van de programma's zijn verkrijgbaar bij ir. P.J.M. Cluitmans. De structuren van de gebruikte database files zijn vermeld in bijlage 1.

Het eerste programma in de boomstructuur is genaamd EMDABS. Hierbij wordt eerst het openingsplaatje getoond, bestaande uit de tekst "EMDABS, Electrophysiologic Monitoring DAtaBase System" en het nummer van het instituut (01). Daarna wordt het hoofdmenu getoond, dat er als volgt uit ziet:

1. ENTER NEW OR EXISTING STUDY CODE
2. STUDY CODE FILE DIRECTORY
3. ACCESS DRUG DATA

In geval 1 wordt het programma EMDAB1 aangeroepen. In geval 2 worden alle studie codes met bijbehorende titel en datum getoond. Deze staan in de database file EMSTUDY.DBF geregistreerd. Er bestaat dan nog de mogelijkheid om een record te veranderen. In dat geval wordt EMHELPER aangeroepen, waardoor een helptekst zichtbaar wordt.



figuur 3.2. dBASE III programma overzicht en de daarbij gebruikte database files.

Omdat hoofdletters en kleine letters verschillende karakters zijn wordt steeds, nadat gegevens ingevoerd of veranderd zijn, een conversie naar hoofdletters verricht. Bij keuze 3 wordt het programma EMDRUGMN aangeroepen.

Bij het EMDAB1 programma worden ook weer alle studie codes met de bijbehorende titel en datum getoond, waarna gevraagd wordt een studie code in te voeren. Deze code bestaat uit maximaal vier karakters, waarbij het eerste karakter een letter moet zijn. Indien de code al bestaat wordt EMCODEMN aangeroepen. Bij een nieuwe code moet de titel van de desbetreffende studie ingevoerd worden, waarna EMABRVCD aangeroepen wordt.

EMABRVCD leidt uit de ingevoerde studie code een unieke code af die uit twee karakters bestaat. Deze code, die ook weer met een letter begint, wordt gebruikt om tijddata files een naam te geven. De afgekorte code wordt in EMSTUDY.DBF opgeslagen en via EMDAB1 wordt weer verder gegaan met EMCODEMN. Omdat bij Oracle de lengte van de tabelnamen in principe niet beperkt is, en dus ook geen afgekorte studie code nodig is, ga ik niet verder in op de gebruikte methode.

Nadat met behulp van de vorige programma's de studie gegevens ingevoerd of veranderd zijn, houdt EMCODEMN zich bezig met de subject codes (subject nivo) en verschijnt het volgende menu:

1. NEW SUBJECT: ASSIGN SUBJECT CODE
2. ENTER EXISTING SUBJECT CODE FOR DATA EDIT/REVIEW
3. REVIEW SUBJECT CODES FOR ALL SUBJECTS IN STUDY ‹studie code›
4. REVIEW DATA FOR ALL SUBJECTS IN STUDY ‹studie code›

Allereerst wordt, indien deze nog niet bestaat, een database file ‹studie code›SESS.DBF aangemaakt met EMSESSDT.DBF als voorbeeld. Bij 1 wordt uitgegaan van de subject code 1001-111-0001 en wordt, zolang deze al in ‹studie code›SESS bestaat, met 1 verhoogd. Als een unieke code gevonden is worden de files ‹studie code›DEMO.DBF en ‹studie code›IDNT.DBF aangemaakt met EMDEMODT.DBF respectievelijk EMIDDATA.DBF als voorbeeld. Dan wordt EMIDMENU aangeroepen om de identificatie gegevens in te voeren.

In geval 2 zorgt EMCODEM1 ervoor dat er een overzicht van de bij de eerder ingevoerde studie code behorende subject codes gegeven wordt. Nadat er dan een subject code ingevoerd is, kijkt dit programma eerst of de code wel goed ingevoerd is, en dan of de code inderdaad al bestaat. Als aan beide voorwaarden voldaan is, wordt er ook verder gegaan met EMIDMENU.

In geval 3 wordt weer een overzicht gegeven van alle bij deze studie behorende subject codes. Dan wordt nog de mogelijkheid geboden van één van deze codes de bijbehorende identificatie gegevens te veranderen, door middel van EMIDMENU.

Bij keuze 4 wordt de subject code gelijk aan "ALL" gemaakt en wordt EMSESSMN aangeroepen. Dit levert echter bij EMSESSMN tot geen enkel resultaat.

Het bij 1 tot en met 3 aangeroepen EMIDMENU maakt het mogelijk om identificatie gegevens toe te voegen, te veranderen of alleen te bekijken. De gegevens zijn of worden opgeslagen in ‹studie code›IDNT.DBF. Het menu van EMIDMENU ziet er als volgt uit:

1. ENTER/EDIT/REVIEW IDENTIFICATION DATA
2. RETURN TO SUBJECT CODE PROGRAM
3. BYPASS IDENTIFICATION PROGRAM

Indien identificatie gegevens gemanipuleerd gaan worden (keuze 1), moet eerst een toegangscode (password) ingevoerd worden. Dan kunnen de naam van de patiënt, ziekenfonds-nummer en het nummer van het medisch dossier ingevoerd of veranderd worden, waarna, net als bij keuze 3, EMDEMOMN aangeroepen wordt.

EMDEMOMN houdt zich bezig met de demografische gegevens, en toont het volgende menu:

1. ENTER/EDIT/REVIEW DEMOGRAPHIC DATA
2. RETURN TO PREVIOUS PROGRAM
3. BYPASS DEMOGRAPHY PROGRAM

Bij keuze 1 krijgt men toegang tot de in ‹studie code›DEMO.DBF opgeslagen gegevens. Dit zijn leeftijd, geslacht, lengte en gewicht van de patiënt, links of rechtshandig, diagnose en eventueel de studie code van een vorige studie. Bovendien bestaat nog de mogelijkheid extra informatie in een zogenaamd memoveld op te slaan. Dan wordt, net als bij keuze 3, verder gegaan met het programma EMSESSMN.

EMSESSMN is het eerste programma dat op sessie nivo werkt. Als bij een gegeven studie en subject nog geen sessie geweest is, krijgt deze sessienummer 01. In het andere geval worden alle bij de subject code behorende sesienummers getoond en kan een al of niet bestaand sessienummer ingevoerd worden. Wanneer er sprake is van een nieuw sessienummer worden in de database file ‹studie code›SESS.DBF de "vlaggen" ECGDBF, APDBF, PA_CVPDBF, VENTDBF, GASDBF, SATDBF, TEMPDBF, EEGDBF, EPDBF en EVENTDBF op "N" gezet, om aan te geven dat al deze database files nog niet bestaan. In een volgende stap kunnen de velden ANESTHETISTS, SURGEONS, TECHNOLOGY en OPERATION

ingevuld worden. Als dit gebeurd is wordt verder gegaan met het EMDABMEN programma.

EMDABMEN en EMDATMEN organiseren zowel de overige sessie-gegevens als de invoer van tijdgegevens. Eerst moet er een klasse gekozen worden, waartoe deze gegevens behoren. Daartoe verschijnt het volgende menu met bij elke klasse de eerdergenoemde "vlag", die aangeeft of er al een database file bestaat:

A. DRUG DATA Y/N
B. EVENT DATA Y/N
C. EVOKED POTENTIAL DATA Y/N
D. EEG DATA Y/N
E. ECG DATA Y/N
F. ARTERIAL PRESSURE DATA Y/N
G. PULMONARY ARTERY/ CVP Y/N
H. GAS ANALYSIS Y/N
I. VENTILATION DATA Y/N
J. OXIMETER/SATURATION DATA Y/N
K. TEMPERATURE DATA Y/N

Wanneer een klasse gekozen is, kan men de bijbehorende sessie-gegevens invoeren of veranderen en files met tijdgegevens construeren. Tenslotte bestaat nog de mogelijkheid, door zelf dBASE III commando's in te voeren, de tijdgegevens te manipuleren. Bij keuze A wordt het programma EMDRUGMN aangeroepen, waar ik zo nog op terug kom. Bij de overige keuzemogelijkheden worden programma's aangeroepen en database files aangemaakt zoals is weergegeven in tabel 3.1.

Tabel 3.1. Overzicht van de door de programma's EMDABMEN en EMDATMEN gebruikte database files en programma's.

KLASSE	SESSIE DATA FILE (.DBF)	VOORBEELD SESSIE DATA FILE (.DBF)	SESSIE PROGRAMMA	EXTENSIE TIJD DATA FILE	VOORBEELD TIJD DATA FILE .DBF
B	<STUDIE CODE>EVNT	EMEVNTSS	EMEVENTS	.EVN	EMEVNDAT
C	<STUDIE CODE>EP	EMEPSESS	EMEPMENU	.EPM	EMEEGDAT
D	<STUDIE CODE>EEG	EMEEGSES	EMEEGMEN	.EPM	EMEEGDAT
E	<STUDIE CODE>ECG	EMECGSES	EMECGMEN	.PEP	EMPRSECG
F	<STUDIE CODE>PRES	EMBPSESS	EMBPMENU	.PEP	EMPRSECG
G	<STUDIE CODE>CVPA	EMPASESS	EMPAMENU	.PEP	EMPRSECG
H	<STUDIE CODE>GAS	EMGASSES	EMGASMEN	.GAS	EMGASDAT
I	<STUDIE CODE>VENT	EMVENTSS	EMVENTMN	.GAS	EMGASDAT
J	<STUDIE CODE>SAT	EMSATSES	EMSATMEN	.GAS	EMGASDAT
K	<STUDIE CODE>TEMP	EMTEMPSS	EMTEMPMN	.TMP	EMTEMPDT

De structuur van de aangemaakte (sessie en tijd) database files wordt steeds gekopieerd van de gegeven voorbeeld files. De bijbehorende "vlag" wordt dan op "Y" gezet. De sessie programma's zorgen ervoor dat de bij die klasse behorende sessiegegevens ingevoerd kunnen worden. De namen van de files met tijdgegevens worden als volgt samengesteld: Het eerste deel van de naam bestaat uit de in EMABRVCD bepaalde

afgekorte studie code. Het tweede deel wordt gevormd door de laatste vier karakters van de subject code. Het derde deel is het sessienummer en het laatste deel is de in de tabel gegeven extensie. Bij een studie code TEST met subject code 1001-111-0001 en sessienummer 01 kan, bij invoer van temperatuurgegevens de volgende filenaam geconstrueerd worden: TE000101.TMP. (Onder MSDOS mag een filenaam uit niet meer dan acht karakters bestaan!).

De tweede tak in de boomstructuur van figuur 3.2 wordt gevormd door de programma's die de gegevens met betrekking tot medicijnen manipuleren. Het EMDRUGMN programma kan zowel vanuit het hoofdmenu (EMDABS) als vanuit het data menu (EMDABMEN) aangeroepen worden. De gegevens zijn opgeslagen in EMDRUGDT.DBF. Het openingsmenu ziet er als volgt uit:

1. FIND DRUG CODE OR NAME
2. EDIT DRUG CODE OR NAME
3. ADD DRUG CODE OR NAME

In geval 1 wordt EMFINDRG aangeroepen. Dit programma zoekt het gevraagde record op door één van de volgende vier gegevens in te voeren: de generieke naam, de handelsnaam, de code van het medicijn of de algemene kategorie waartoe het medicijn behoort. Bij dit laatste kan de keuze gemaakt worden uit de volgende kategorieën:

A. ANTICHOLINERGICS
B. BETA BLOCKERS
C. CALCIUM CHANNEL BLOCKERS
D. DIURETICS
E. ANTICHOLINESTERASES
G. GASES
H. HYPNOTICS/SEDATIVES
I. INTRAVENOUS ANESTHETICS
K. COAGULATION
L. LOCAL ANESTHETICS
M. MUSCLE RELAXANTS
N. NARCOTICS
O. VASODILATORS
P. PRESSORS/INOTROPES
Q. COLLOIDS
R. RESPIRATOR
S. STEROIDS
T. TRANQUILIZERS/ANTIEMETICS
V. VOLATILE ANESTHETICS
W. ANTIBIOTICS
X. ANTICONVULSANTS
Y. ANTACIDS/GI
Z. OTHER

De code van het medicijn bestaat uit een letter die de klasse vertegenwoordigt en daarachter drie cijfers. Bij keuze-mogelijkheid 2 wordt EMEDITDR aangeroepen. Hierbij gebeurt hetzelfde als bij 1, alleen kan nu een gevonden record ook veranderd worden.

Wanneer men nummer 3 gekozen heeft wordt EMADDRUG aangeroepen. Dit programma vraagt eerst om de klasse waartoe het toe te voeren medicijn behoort. Daarbij wordt weer het bovengenoemde overzicht van de diverse klassen gegeven. Nadat de klasse gekozen is moet de generieke naam ingevoerd worden. De code wordt nu als volgt toegewezen: eerst wordt de plaats

van de generieke naam binnen een klasse bepaald door ze alfabetisch te ordenen. De nieuwe code wordt dan gevormd door het gemiddelde te nemen van de code van het voorgaande medicijn en het medicijn wat erna komt. Indien de nieuwe generieke naam laatste in de rij is, wordt bij de grootste code binnen die klasse 10 opgeteld. Een voorbeeld: stel we willen halothaan toevoegen aan de gassen lucht en helium. Het resultaat komt er dan als volgt uit te zien:

```
G100 AIR
G105 HALOTHANE
G110 HELIUM
```

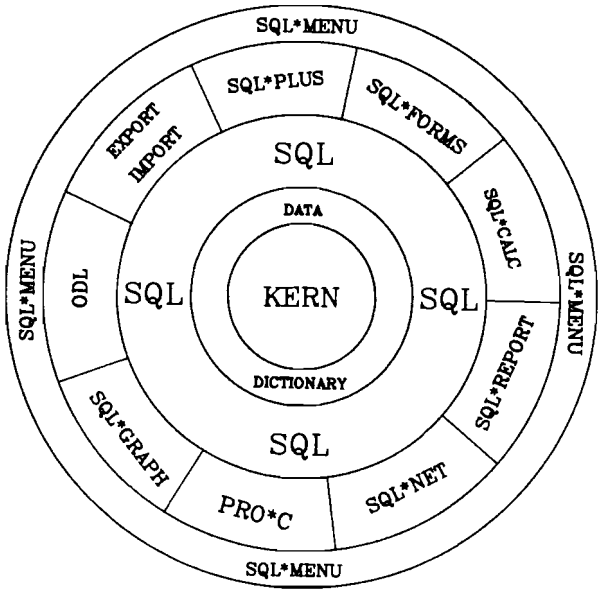
Als de code bekend is bestaat nog de mogelijkheid maximaal twee handelsnamen toe te voegen. Opgemerkt dient te worden dat bij deze manier van coderen slechts een beperkt aantal medicijnen toegevoegd kan worden.

Uit de voorgaande beschrijving van de verzameling programma's waaruit EMDABS is opgebouwd, blijkt al dat een optimalisatie van het geheel mogelijk is. Zo is het bijvoorbeeld gunstig een aantal programma's samen te voegen tot één. Voorbeelden hiervan zijn: EMDABS en EMDAB1; EMCODEMN en EMCODEM1; EMFINDRG, EMEDITDR en EMADDRUG. Mijn ervaring bij het gebruik van deze EMDABS is, dat het aantal database files, dat tijdens een run dynamisch gecreëerd wordt, nogal omvangrijk is, zodat het overzicht snel verloren gaat. Ik denk dat het daarom ook beter is verschillende kleine, gelijkvormige database files onder te brengen in een grotere file. Meer hierover in hoofdstuk 5.

4 ORACLE

4.1 Overzicht

Het relationele database management systeem (RDBMS) Oracle is door Relational Software Inc. ontwikkeld. De eerste versie verscheen al in 1979. Inmiddels zijn er versies voor praktisch alle soorten computers beschikbaar, van mainframe tot micro. Het Oracle systeem kan schematisch als volgt worden voorgesteld (Lit. 1,5):



figuur 4.1. Overzicht van het RDBMS Oracle.

In het midden bevindt zich de kern. Deze bevat alle programmatuur die nodig is voor het beheer van de database en de andere systeemonderdelen. De data dictionary is een verzameling tabellen, die Oracle zelf gebruikt om bijvoorbeeld gebruikerscodes en de bijbehorende privileges en tabelnamen in op te slaan. Daar omheen bevindt zich als het ware een schil die uit SQL bestaat. Alle opdrachten die gegeven worden aan de kern moeten "door" deze schil. Dat wil zeggen dat het alleen mogelijk is opdrachten aan de kern te formuleren in SQL! Ook de hulpmiddelen, die in de volgende schil zijn weergegeven, communiceren alleen via SQL met de database. Deze hulpmiddelen zijn onderverdeeld in de Oracle Tools en de Oracle Utilities.

De Oracle Tools zijn softwarepakketten die afzonderlijk geïnstalleerd kunnen worden. Dit zijn SQL*PLUS, SQL*FORMS, SQL*CALC, SQL*REPORT, SQL*NET, PRO*C, SQL*MENU en SQL*GRAPH.

SQL*PLUS is het interaktieve SQL interface. Door middel van SQL commando's, welke in de volgende paragraaf besproken

worden, kan men de database direkt raadplegen. Bovendien biedt SQL*PLUS nog extra mogelijkheden met betrekking tot het veranderen en opslaan van ingevoerde commando's en het afdrukken van gegevens in een eenvoudig formaat.

SQL*FORMS is een interaktieve applicatie generator. Zonder het schrijven van een programma is het mogelijk applicaties te ontwerpen, waarmee je gemakkelijk gegevens kunt toevoegen, selecteren, veranderen en verwijderen. Via verschillende menu's en een zogenaamde "screen painter" kunnen de eisen, die aan de applicatie gesteld worden, kenbaar gemaakt worden.

SQL*CALC is een spreadsheet dat grote overeenkomsten vertoont met Lotus 1-2-3. Via SQL*CALC is het mogelijk zowel gegevens die met bulp van de spreadsheet zijn ingevoerd in de database op te slaan, als informatie uit de database in spreadsheet-vorm weer te geven.

Met behulp van SQL*REPORT kunnen gegevens uit de database, volgens een opgegeven formaat weergegeven worden.

SQL*NET stelt je in staat om toegang te krijgen tot een Oracle database systeem, dat op een andere computer draait.

Door gebruik te maken van de precompiler PRO*C wordt het mogelijk programma's te schrijven in de procedurele programmeertaal C, die de gegevens in de Oracle database kunnen manipuleren. PRO*C onder PC/MS-DOS ondersteunt de Lattice en Microsoft C-compilers. Precompilers voor de programmeertalen cobol, fortran, PL/I, pascal en ada zijn ook verkrijgbaar.

Met SQL*MENU kunnen menu's gemaakt worden van waaruit verschillende programma's aangeroepen worden. SQL*GRAPH stelt je in staat database informatie grafisch weer te geven. De laatste twee tools zijn voor de PC-versie nog niet beschikbaar.

De belangrijkste Oracle Utilities zijn Export (EXP), Import (IMP) en de Oracle Data Loader (ODL). Met de export faciliteit kan een backup van (een deel van) de database gemaakt worden. De import faciliteit voert de geëxporteerde data weer in de database in. Met behulp van de Oracle Data Loader kunnen grote hoeveelheden (ASCII) gegevens in de Oracle database ingelezen worden.

4.2 SQL

SQL (uitgeproken als "sequel") betekent Structured Query Language en is ontwikkeld door IBM. Het is een vierde-generatietaal, dat wil zeggen dat SQL niet procedureel is, zoals bij derde-generatietalen het geval is. Om dit toe te

lichten wordt in het onderstaande voorbeeld SQL vergeleken met de derde-generatietaal pascal. In beide programmastukjes worden die rijen uit de studenttabel geselecteerd, waarvoor geldt dat de woonplaats Eindhoven is.

PASCAL:

```
TYPE rec=record
     stud_id, naam, plaats: string[25]
     END;
VAR  F: file of rec; buff: rec;
BEGIN assign(F:'student'); reset(F);
      WHILE NOT EOF (F) DO
      BEGIN READ(F,buff);
            IF buff.plaats='EINDHOVEN' THEN
            BEGIN WRITE buff.stud_id;
                  WRITE buff.naam;
                  WRITE buff.plaats
            END
      END
END.
```

SQL:

```
SELECT stud_id, naam, plaats
FROM   student
WHERE  plaats='EINDHOVEN'
```

Uit dit voorbeeld blijkt dat men in SQL het gewenste resultaat definieert (het WAT) en niet de te volgen procedure (het HOE). SQL kan onderverdeeld worden in vier delen (Lit 1,4):

1. Queries, ofwel het opvragen van gegevens uit de tabellen. Queries beginnen altijd met de speciaal daarvoor gereserveerde SQL opdracht SELECT.

2. Data Manipulation Language (DML).
   Zoals al in hoofdstuk 2 gezegd is, kunnen we hiermee gegevens toevoegen, wijzigen en verwijderen. De bijbehorende commando's zijn INSERT, UPDATE en DELETE.

3. Data Definition Language (DDL).
   Hiermee kunnen we database objecten creëeren, veranderen of verwijderen. De DDL commando's zijn CREATE, ALTER en DROP.

4. Data Control Language (DCL).
   Met de DCL kan een gebruiker een andere gebruiker de toegang tot zijn gegevens verlenen of kan de toegewezen bevoegdheden weer ontzeggen. De belangrijkste DCL commando's zijn GRANT en REVOKE.

De volgende vier paragrafen behandelen kort de meest voorkomende SQL commando's. Daarbij is de bovenstaande

indeling aangehouden.

4.2.1 Het raadplegen van gegevens

Voor het opvragen (presenteren) van gegevens uit de database is in SQL één commando aanwezig, namelijk SELECT. De basisstructuur ziet er als volgt uit:

```
selecteer kolommen
van tabel of tabellen
waarbij elke rij voldoet aan...
```

Het resultaat van een selectie wordt de resultaattabel genoemd. Met het "*" karakter kunnen we alle kolommen selecteren, dus met

```
SELECT *
FROM   tabelnaam
```

wordt de gehele tabel gepresenteerd. Oracle slaat alle tabelnamen op in de tabel genaamd TAB. Het commando

```
SELECT *
FROM   TAB
```

laat dus alle aan de gebruiker toebehorende tabellen zien. Het is ook mogelijk op één of meerdere kolommen te sorteren. Daartoe het select commando uitgebreid met ORDER BY. De basisvorm wordt dan:

```
SELECT kolomnaam 1 (, kolomnaam 2,...)
FROM   tabelnaam
ORDER BY kolomnaam 1 (ASC/DESC)
       (,kolomnaam 2 (ASC/DESC),...)
```

Met ASC (ascending) en DESC (descending) kan per kolom aangegeven worden of opklimmend of afdalend gesorteerd moet worden. Bij weglaten van ASC en DESC wordt opklimmend gesorteerd. Merk op dat met SELECT FROM een eerdergenoemde projektie gerealiseerd wordt. Het selecteren van rijen gebeurt met de WHERE clausule en heeft de volgende basisvorm:

```
WHERE voorwaarde 1 (logische operator voorwaarde 2 ...)
```

Een WHERE clausule bestaat dus uit één of meerdere voorwaarden die met elkaar verbonden worden door logische operatoren (AND, OR en NOT). Een voorwaarde bestaat steeds uit twee expressies en een operator (=, >, >=, < of <=) die de relatie tussen de twee expressies weergeeft. In het volgende voorbeeld worden de registratienummers van die studenten in de studenttabel geselecteerd, die de naam Jansen hebben en in Eindhoven wonen:

```
SELECT stud_id
FROM   student
WHERE  naam='JANSEN' AND plaats='EINDHOVEN'
```

Het verbinden van twee of meer tabellen kan gerealiseerd worden met behulp van de volgende JOIN constructie:

```
SELECT kolom a, kolom b,...
FROM   tabelnaam 1, tabelnaam 2,...
WHERE  voorwaarde(n)
```

Wanneer we bijvoorbeeld de namen van de studenten willen weergeven met de behaalde resultaten, dan kunnen de tabellen student en resultaat (zie paragraaf 2.4) als volgt samengevoegd worden:

```
SELECT naam, vak_id, cijfer
FROM   student, resultaat
WHERE  student.stud_id=resultaat.stud_id
```

Bij de JOIN worden dus verschillende kolommen uit verschillende tabellen samengevoegd. Het is in SQL ook mogelijk om rijen uit verschillende tabellen samen te voegen. Dit gebeurt met behulp van het commando UNION:

```
SELECT
FROM   tabelnaam 1
WHERE
     UNION
          SELECT
          FROM   tabelnaam 2
          WHERE
               UNION ...
```

4.2.2 Het manipuleren van gegevens

Nadat een tabel gecreëerd is, kan deze gevuld worden met behulp van het INSERT command. We kunnen hierbij twee situaties onderscheiden, namelijk de gegevens worden vanuit een andere tabel ingevoerd of de gegevens worden expliciet verstrekt. In het eerste geval maakt men gebruik van de volgende basisconstructie:

```
INSERT INTO tabelnaam 1 (kolomnaam 1, kolomnaam 2,...)
     SELECT kolommen
     FROM   tabelnaam 2
    (WHERE  voorwaarde)
```

In het tweede geval wordt de volgende constructie gebruikt:

```
INSERT INTO tabelnaam (kolomnaam 1, kolomnaam 2,...)
     VALUES (gegeven 1, gegeven 2,...)
```

Naast het toevoegen van rijen kan ook de bestaande inhoud van een tabel gewijzigd worden. Het veranderen van individuele gegevens kan gebeuren met behulp van het UPDATE TABLE commando:

```
UPDATE TABLE tabelnaam
         SET kolomnaam 1 = expressie
          (, kolomnaam 2 = expressie ...)
        (WHERE voorwaarde)
```

Voor iedere rij waarbij het resultaat van de WHERE clausule "waar" is, worden de veranderingen, die gespecificeerd zijn achter SET, uitgevoerd. Wil men complete rijen uit een tabel verwijderen, dan kan dat worden gerealiseerd met het DELETE commando. De basisvorm ziet er als volgt uit:

```
DELETE FROM tabelnaam
(WHERE voorwaarde)
```

Met de WHERE clausule wordt aangegeven welke rij of rijen uit de tabel moeten worden verwijderd. Zonder deze clausule worden alle rijen uit de genoemde tabel worden verwijderd. De tabeldefinitie blijft echter wel bestaan.

Met behulp van het commando COMMIT worden alle veranderingen definitief gemaakt. Door het ROLLBACK commando te gebruiken worden alle veranderingen, die nog niet definitief gemaakt zijn, ongedaan gemaakt.

4.2.3 Het definiëren van database objecten

Het SQL commando voor het definiëren van tabellen is CREATE:

```
CREATE TABLE tabelnaam
  (kolomnaam 1 gegevenstype (NOT NULL),
   kolomnaam 2 gegevenstype (NOT NULL),...)
```

Voor het gegevenstype kan een keuze gemaakt worden uit karakter, numeriek of datum. Wanneer de kwalificatie NOT NULL wordt toegevoegd, is het vullen van deze kolom voor iedere rij van de tabel verplicht. Een primaire sleutel van een tabel heeft dus altijd de NOT NULL kwalificatie. Een bestaande tabel wordt als volgt met behulp van het DROP commando verwijderd:

```
DROP TABLE tabelnaam
```

Voor het toevoegen van kolommen aan de tabel heeft SQL het ALTER commando:

```
ALTER TABLE tabelnaam
        ADD (kolomnaam 1 gegevenstype, kolomnaam 2 ...)
```

De nieuwe kolom mag niet de kwalificatie NOT NULL hebben, omdat in de al bestaande rijen van de tabel NULL waarden onstaan. Het verwijderen van kolommen kan in principe worden gerealiseerd volgens de volgende methode:

1. Definieer een nieuwe tabel die alle kolommen van de oude tabel bevat, behalve de te verwijderen kolom(men).
2. Vul de nieuwe tabel met de gegevens van de oude tabel met behulp van het INSERT commando.
3. Verwijder de oude tabel met het DROP TABLE commando.

De rijen in een tabel zijn in een willekeurige volgorde geplaatst. Nieuwe rijen worden toegevoegd aan het einde van een tabel. Bij zoekprocessen kan de tijd, die nodig is voor het doorlopen van een tabel, flink oplopen. Wanneer de inhoud van een of meer kolommen van een zogenaamde index wordt voorzien, kan dit zoekproces worden bekort, zodat een vraag (SELECT) sneller beantwoord kan worden. Naast dit voordeel hebben indexen ook een nadeel. Bij het wijzigen van een tabelinhoud door INSERT, UPDATE of DELETE worden ook telkens de eventuele indexen van deze tabel aangepast, wat aanleiding kan geven tot langere verwerkingstijden (lit. 10). Daarom zal het toepassen van indexen op tabellen afhankelijk zijn van de omstandigheden waaronder de tabellen gebruikt worden:

-Wanneer het selecteren van gegevens de meest voorkomende bewerking is, en er dus weinig wijzigingen aangebracht worden, is het gebruik van indexen aan te bevelen. Indexeer echter op nooit meer dan twee of drie kolommen.
-Bij veel veranderingen van de tabelinhoud van kleine tabellen is het gebruik van indexen minder gewenst.

In tegenstelling tot dBASE III hoeft in SQL niet expliciet aangegeven te worden dat een index gebruikt moet worden. Een index wordt als volgt gedefinieerd:

```
CREATE (UNIQUE) INDEX indexnaam
       ON  tabelnaam (kolomnaam 1, kolomnaam 2,...)
```

Het gebruik van de UNIQUE optie heeft tot gevolg dat in de genoemde kolom(men) alleen unieke gegevens of combinatie van gegevens voor mogen komen. Evenals voor tabellen kan het DROP commando ook voor het verwijderen van indexen gebruikt worden:

```
DROP INDEX indexnaam
```

Als een tabel wordt verwijderd, wordt automatisch ook de bijbehorende index verwijderd. Zoals is vermeld bij het SELECT comando is het resultaat van een SELECT weer een tabel. Deze tabel is normaal gesproken alleen tijdelijk beschikbaar op het beeldscherm of printerpapier. Er vindt

geen permanente opslag van deze tabel plaats. Met behulp van een zogenaamde View (virtuele tabel) kan het resultaat van een SELECT commando worden opgeslagen onder een bepaalde naam. Men heeft dan de beschikking over een nieuwe tabel. Een View wordt als volgt gedefinieerd:

```
CREATE VIEW viewnaam (kolomnaam 1, kolomnaam 2,...)
            AS SELECT commando
```

Views kunnen als normale tabellen worden gebruikt. Het is zelfs toegestaan om weer een nieuwe view te baseren op een bestaande view. Met behulp van het DROP commando kan een view weer uit de database worden verwijderd:

```
DROP VIEW viewnaam
```

4.2.4 De autorisatie van het gebruik

In een omgeving waarin meerdere gebruikers gelijktijdig of achtereenvolgens gebruik maken van dezelfde gegevens, moeten maatregelen kunnen worden genomen voor ket goed laten verlopen van dit gemeenschappelijk gebruik. Iedere gebruiker van een DBMS moet bepaalde bevoegdheden hebben die betrekking hebben op de toegang tot de database en het mogen uitvoeren van bepaalde bewerkingen op tabellen. Deze bevoegdheden worden toegekend met behulp van het GRANT commando:

```
GRANT bevoegdheid 1 (, bevoegdheid 2,...)
(ON    object 1 (, object 2,...))
TO     gebruiker 1 (, gebruiker 2,...)
IDENTIFIED BY toegangscode
```

De default waarde voor het object is "SYSTEM", zodat de ON clausule daarbij achterwege kan blijven. Er kan dan gekozen worden uit de bevoegdheden DBA (alle bevoegdheden), CONNECT (alleen SELECT bevoegdheid) en RESOURCE (alle bevoegdheden, behalve gebruikers definiëren). Wanneer men als object een of meerdere tabelnamen invult, kunnen de volgende bevoegdheden daaraan gegeven worden: ALTER, DELETE, INSERT, SELECT, UPDATE of ALL PRIVILEGES. Voor het implementeren van EMDABS heb ik als volgt de gebruikerscode "emdabs" met de toegangscode "holland" ingevoerd:

```
GRANT RESOURCE TO emdabs IDENTIFIED BY holland
```

Eenmaal verleende bevoegdheden kunnen worden ontnomen met behulp van REVOKE:

```
REVOKE bevoegdheid 1 (, bevoegdheid 2,...)
ON     object 1 (, object 2,...)
FROM   gebruiker 1 (, gebruiker 2,...)
```

4.3 SQL*PLUS

De in paragraaf 4.2 behandelde SQL commando's kunnen alleen via SQL*PLUS interaktief gebruikt worden. Dus de Oracle database gebruiken kan alleen via SQL en SQL interaktief gebruiken kan alleen via SQL*PLUS. Voor een uitgebreide behandeling van alle te gebruiken (SQL*PLUS) commando's wordt verwezen naar (Lit. 9). SQL*PLUS wordt opgestart door het commando SQLPLUS. Voorwaarde is wel dat het oracle systeem al aktief is (zie paragraaf 4.7). Net zoals bij alle andere Oracle hulpmiddelen, krijgt de gebruiker pas toegang tot SQL*PLUS nadat een juiste gebruikerscode met bijbehorende toegangscode (password) is ingevoerd. Merk op dat een password met het in paragraaf 4.2.4 behandelde GRANT commando veranderd kan worden.

Eenmaal ingelogd kunnen de SQL commando's regel voor regel ingevoerd worden. Door achter het laatste commando een ";" te plaatsen wordt de reeks van commando's uitgevoerd. Het commando RUN (of R) voert de ingevoerde commando's nog een keer uit. Ingevoerde commando's kunnen met behulp van SAVE in een file met de extensie .sql opgeslagen worden. START ‹filenaam› of @‹filenaam› voert de in een file opgeslagen commando's uit. Een foutief ingevoerd commando kan met behulp van het CHANGE commando C/oud/nieuw verbeterd worden. Binnen SQL*PLUS kunnen ook operating system commando's uitgevoerd worden door er een "$" voor te zetten. Met behulp van EXIT verlaat men SQL*PLUS.

Voordat men met SQL*FORMS een applicatieprogramma maakt, moeten eerst de tabellen, die in de applicatie gebruikt worden, met behulp van het CREATE commando binnen SQL*PLUS gedefinieerd worden. Omdat EMDABS met behulp van SQL*FORMS ontworpen wordt, ga ik daar nu dieper op in.

4.4 SQL*FORMS

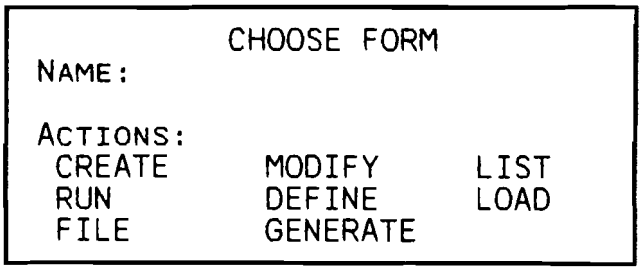
Met SQL*FORMS kunnen applicaties ontwikkeld worden, waarmee op een eenvoudige manier gegevens toegevoegd, opgevraagd, veranderd en verwijderd kunnen worden. De Interactive Application Generator (IAG) compileert het met behulp van SQL*FORMS gegenereerde applicatieprogramma en de Interactive Application Processor (IAP) genaamd RUNFORM runt uiteindelijk de applicatie.

De applicaties die met behulp van SQL*FORMS ontwikkeld worden, zijn gebaseerd op zogenaamde forms. Een form is een rangschikking van gegevens op het scherm van de computer. De gegevens kunnen één scherm of pagina in beslag nemen of meerdere. Elke pagina bestaat uit een of meer blokken, waarbij elk blok naar een bijbehorende basistabel kan verwijzen. De velden die in een blok gedefinieerd worden,

worden gebruikt om gegevens weer te geven, in te voeren of te veranderen. Als een blok een basistabel heeft, komen de velden op het scherm overeen met de velden in de basistabel. Een applicatie kan opgebouwd zijn uit meerdere forms.

SQL*FORMS wordt gestart door middel van het commando SQLFORMS, waarna weer een gebruikerscode en een toegangscode ingevoerd moeten worden. Het ontwerpen van een form verloopt menugestuurd. Er zijn in totaal 25 verschillende menu's die gebruikt kunnen worden. In bijlage 3 is een overzicht van alle menu's gegeven. De meest belangrijke menu's zal ik nu bespreken. Voor een uitgebreide dokumentatie wordt verwezen naar naar (Lit. 7,8).

Het eerste menu dat verschijnt bij het gebruik van SQL*FORMS is het CHOOSE FORM menu (zie figuur 4.2). Hierin wordt de naam van de form ingevoerd, waarmee gewerkt gaat worden.



figuur 4.2. Het CHOOSE FORM menu

Zoals uit figuur 4.2 blijkt, kunnen een aantal akties ondernomen worden. Dit gebeurt door met de cursor op de betreffende aktie te gaan staan en dan de [SELECT] toets in te drukken. Steeds wanneer een bepaalde aktie ondernomen is, kan deze met de [ACCEPT] toets afgesloten worden. Een aktie kan ongedaan gemaakt worden met de [EXIT/CANCEL] toets. (Toets F8 geeft op elk moment een overzicht van de mogelijke funktietoetsen).

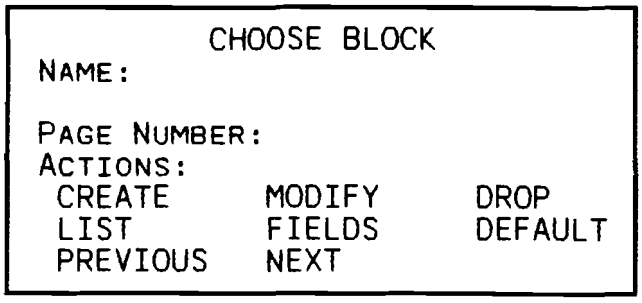
Met de aktie CREATE wordt een nieuwe form gecreëerd en MODIFY stelt je in staat een bestaande form te veranderen. LIST geeft een overzicht van de bestaande forms. De RUN aktie heeft bij de PC-versie van Oracle geen betekenis. Met DEFINE kan een bestaande formnaam veranderd worden en kan een trigger ingevoerd worden.

Een trigger is een stukje programma dat wordt uitgevoerd bij het plaatsvinden van een bepaalde gebeurtenis tijdens een run. Voorbeelden van zo'n gebeurtenis zijn het indrukken van een funktietoets, het toevoegen van records aan de database en naar een volgend blok gaan. Een trigger kan op drie nivo's worden aangemaakt, namelijk form nivo, blok nivo en veld

nivo. Een form nivo trigger is in de gehele form (dus in alle blokken en velden) aktief. Een trigger op blok nivo kan alleen geaktiveerd worden door gebeurtenissen binnen een blok. Het veld nivo is het laagste nivo waarop een trigger gedefinieerd kan worden. Deze trigger heeft dus alleen betrekking op dat ene veld, waarbij hij gedefinieerd is.

Met FILE kunnen veranderingen aan de form bewaard of verwijderd worden, kan de form gekopieerd en hernoemd worden of kan de gehele form uit de database verwijderd worden. De LOAD aktie maakt het mogelijk forms weer in te laden die met behulp van de FILE aktie geheel verwijderd zijn. De GENERATE aktie maakt een output file aan. Meer hierover in paragraaf 4.4.2.

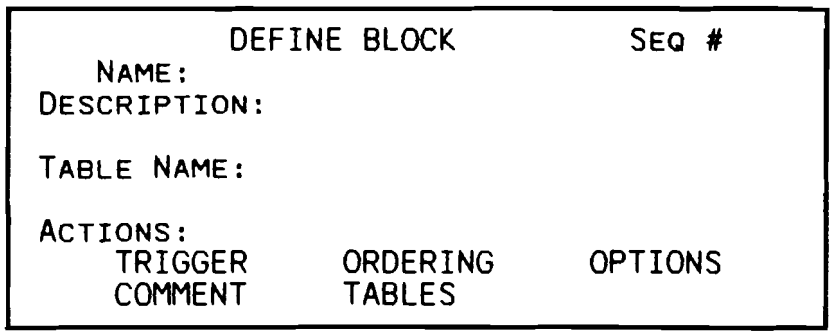
Wanneer de aktie CREATE of MODIFY gekozen is, wordt het CHOOSE BLOCK menu zichtbaar (zie figuur 4.3). Zoals al eerder vermeld is, vormen de blokken de eenheden waaruit een form is opgebouwd.



figuur 4.3. Het CHOOSE BLOCK menu.

Met de akties CREATE en MODIFY kan respectievelijk een nieuw blok aangemaakt of een bestaand blok veranderd worden. In beide gevallen komt men in de zogenaamde "screen painter" terecht. Hiermee kunnen velden op een willekeurige plaats op het scherm (pagina) "neergelegd" worden en kan begeleidende tekst toegevoegd worden. Een andere manier om een nieuw blok aan te maken is met behulp van de DEFAULT aktie mogelijk. In dit geval worden de velden automatisch over de pagina verdeeld. Deze indeling kan dan weer met behulp van MODIFY gewijzigd worden. LIST geeft een overzicht van de bestaande blokken en met DROP kan een blok verwijderd worden. De aktie FIELDS geeft een overzicht van de bij het blok gedefinieerde velden.

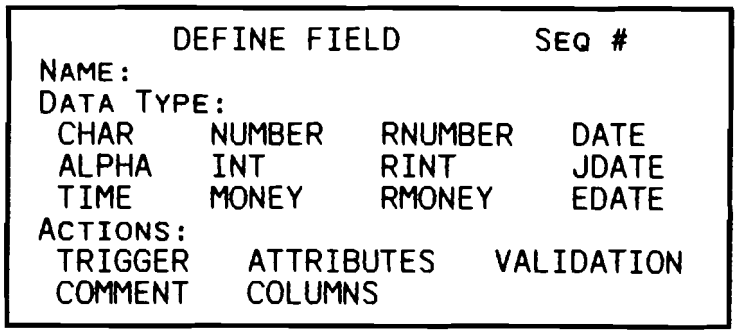
Als we met de screen painter een blok aan het veranderen zijn, kan door de toetsen [SELECT BLOCK] en [DEFINE] na elkaar in te drukken, het DEFINE BLOCK menu zichtbaar gemaakt worden (zie figuur 4.4).



figuur 4.4. Het DEFINE BLOCK menu.

In dit menu kan een korte beschrijving van het betreffende blok ingevoerd worden en kan de naam van het blok veranderd worden. Als een blok gekoppeld is aan een tabel, wordt in dit menu ook de naam van deze tabel ingevoerd. Met de aktie TRIGGER kan een trigger op blok nivo aangemaakt worden. De aktie ORDERING maakt het mogelijk om een ORDER BY en/of een WHERE clausule in te voeren, waardoor de records zowel geordend als geselecteerd kunnen worden. Met OPTIONS kan bijvoorbeeld het aantal rijen dat weergegeven moet worden, opgegeven worden. TABLES geeft een overzicht van de in de database aanwezige tabellen.

Het DEFINE FIELD menu (zie figuur 4.5) verschijnt wanneer de toets [DEFINE] ingedrukt wordt met de cursor op een veld of wanneer met behulp van de [CREATE FIELD] toets een nieuw veld gedefinieerd wordt.

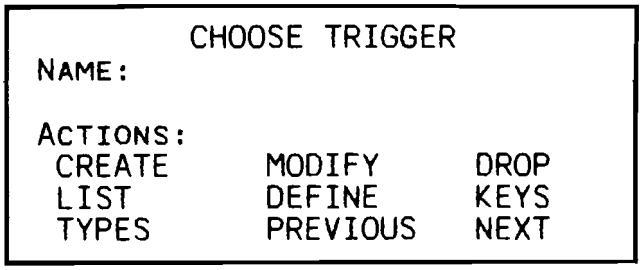


figuur 4.5. Het DEFINE FIELD menu.

Met dit menu kan de veldnaam en het datatype van het veld ingevoerd of veranderd worden. Het aanmaken van een trigger op veld nivo gebeurt met de TRIGGER aktie. Door de aktie ATTRIBUTES te kiezen, wordt het mogelijk de attributen van het veld te specificeren, zoals bijvoorbeeld of het veld een database veld is, of het zichtbaar moet zijn, of invoer toegestaan is, of invoer verplicht is en of er een

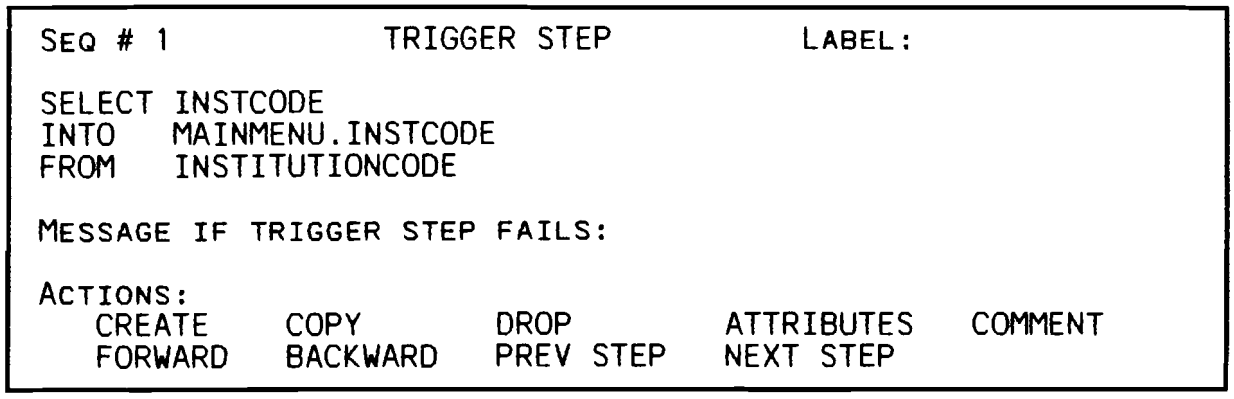
automatische conversie naar hoofdletters moet plaatsvinden. Met de VALIDATION aktie kan bijvoorbeeld opgegeven worden tussen welke waarden de invoer moet liggen en wat de default waarde van een veld is. COLUMNS laat de namen van alle kolommen zien van de bij het blok behorende tabel.

In de voorgaande menu's is steeds gesproken van het aanmaken van triggers met behulp van de TRIGGER aktie. Het menu dat bij deze aktie hoort ziet er als volgt uit:



figuur 4.6. Het CHOOSE TRIGGER menu.

Als naam van de trigger wordt normaal gesproken het type van de trigger ingevoerd. De aktie TYPES geeft een overzicht van de triggertypes op het huidige nivo (form, blok of veld). Voorbeelden van triggertypes zijn PRE-FIELD, POST-FIELD, POST-CHANGE en KEY triggers. Van de laatste soort triggers wordt met behulp van de aktie KEYS een overzicht gegeven. Na de aktie CREATE (of MODIFY) gekozen te hebben, verschijnt het trigger step menu (zie figuur 4.7).



figuur 4.7. Het TRIGGER STEP menu.

Een trigger kan uit meerdere stappen opgebouwd zijn. Elke stap van de trigger kan slagen of mislukken. Met behulp van de ATTRIBUTES aktie kan aangegeven worden met welke stap verder gegaan moet worden in geval van slagen en in geval van mislukken. Per stap kan een van de volgende commando's

ingevoerd worden:

1. Een SQL commando (zie paragraaf 4.2).
   Om een uitwisseling van gegevens tussen de velden op het scherm en de velden in een tabel mogelijk te maken, is het SELECT commando uitgebreid met de INTO clausule:

```
SELECT kolomnaam
INTO   bloknaam.(scherm)veldnaam
FROM   tabelnaam
WHERE  voorwaarde
```

2. Een SQL*FORMS commando, waarmee het indrukken van een funktietoets gesimuleerd kan worden. De volgende trigger bijvoorbeeld verplaatst de cursor naar het volgende blok:

```
#EXEMACRO NXTBLK;
```

3. Een user exit, waarmee bijvoorbeeld een programma aangeroepen kan worden, dat in de programmeertaal C geschreven is.

Zoals al eerder gezegd is kan een applicatie uit meerdere forms opgebouwd zijn. Bij het runnen van een applicatie kan vanuit een form een andere form aangeroepen worden. Nadat de forms met behulp van SQL*FORMS zijn ontwikkeld, moeten ze uitvoerbaar gemaakt worden. De eerste stap vormt het uitvoeren van de GENERATE aktie in het SQL*FORMS CHOOSE FORM menu. Hierdoor wordt een file genaamd ‹formnaam›.INP aangemaakt. Dit is een leesbare tekstfile, waarin de form in een vraag-en-antwoord vorm beschreven wordt. De volgende stap is het compileren van de INP file met behulp van de Interactive Application Generator (IAG). Het commando is als volgt:

```
IAG formnaam
```

De IAG produceert een file genaamd ‹form›.FRM. In de laatste stap wordt een form werkelijk uitgevoerd: RUNFORM leest de FRM file in en voert deze uit. Het commando luidt:

```
RUNFORM form gebruikerscode/toegangscode
```

4.5 De export en import faciliteiten

De Oracle database is opgebouwd uit een beperkt aantal partities, welke overeenkomen met fysieke files. Als gegevens aan de database toegevoegd worden, worden ze dus in één van deze partities opgeslagen. Tabelnamen worden dan ook niet in de directory waargenomen, wat overigens bij dBASE III wel het geval is.

Om nu toch van tabellen een backup in filevorm te kunnen maken, wordt de export faciliteit gebruikt (Lit. 6). Met behulp van de import faciliteit kan de backup weer in de database opgenomen worden. Het is op deze manier ook mogelijk gegevens tussen de databases van verschillende computer-systemen uit te wisselen. Bij het exporteren van gegevens kan uit de volgende drie mogelijkheden een keuze gemaakt worden:

1. alleen tabellen exporteren (table mode).
2. tabellen, indexen en bevoegdheden exporteren (user mode).
3. de gehele database exporteren (full database mode).

De laatste keuze is alleen mogelijk voor de systeembeheerder (DataBase Administrator, DBA). Het export commando luidt:

```
EXP gebruikerscode/toegangscode
```

Na het EXP commando uitgevoerd te hebben worden nog enkele vragen gesteld, zoals bijvoorbeeld in welke mode geëxporteerd gaat worden.

De import faciliteit werkt op dezelfde manier. Het import commando luidt:

```
IMP gebruikerscode/toegangscode
```

4.6 De Oracle Data Loader (ODL)

ODL is een op zichzelf staande faciliteit, die wordt gebruikt om grote hoeveelheden gegevens uit files met operating system formaat (ASCII) in de Oracle database te laden. EMDABS gebruikt ODL om op grote schaal tijdgegevens in te voeren. Bij ODL spelen de volgende files een rol (Lit 6):

control file	:	Deze file specificeert de data file en de tabellen met bijbehorende kolommen, waarop de gegevens afgebeeld gaan worden.
data file	:	Deze file bevat de data die moet worden ingeladen.
log file	:	Hierin worden het aantal gelezen, geladen en verworpen records vermeld en de eventuele foutboodschappen.
bad file	:	Hierin worden de records opgeslagen, die niet door de database geaccepteerd zijn, bijvoorbeeld wanneer karakters in een numeriek veld geplaatst worden.

Het ODL commando ziet er als volgt uit:

```
ODL control-filenaam log-filenaam gebruikerscode/toegangscode
```

Ik heb de Oracle Data Loader gebruikt om de gegevens met betrekking tot de medicijnen vanuit dBASE III naar Oracle over te brengen. Daartoe heb ik eerst de database file EMDRUGDT.DBF omgezet in een ASCII file genaamd DRUGS.DAT, welke daarna met ODL in de Oracle tabel DRUGDATA is ingeladen. De gebruikte control file DRUG.CTL is hieronder weergegeven.

```
DEFINE RECORD DRUGDATAREC AS
   DRUGCODE (CHAR(4),LOC(+1)),
   GENERICN (CHAR(26),LOC(+1)),
   TRADEN1  (CHAR(16),LOC(+1)),
   TRADEN2  (CHAR(14),LOC(+1));
DEFINE SOURCE FILE
   FROM DRUGS.DAT
   LENGTH 64
   CONTAINING DRUGDATAREC;
FOR EACH RECORD
   INSERT INTO DRUGDATA
   (DRUG_CODE,GENERIC_N,TRADE_N1,TRADE_N2) VALUES
   (DRUGCODE,GENERICN,TRADEN1,TRADEN2)
NEXT RECORD
```

4.7 De installatie van Oracle

Om Oracle versie 5.1 op een PC te installeren worden de volgende eisen gesteld:

- Een IBM Personal Computer AT of een 100% compatibel computer.
- 640 Kb RAM.
- Minstens 896 Kb extended memory

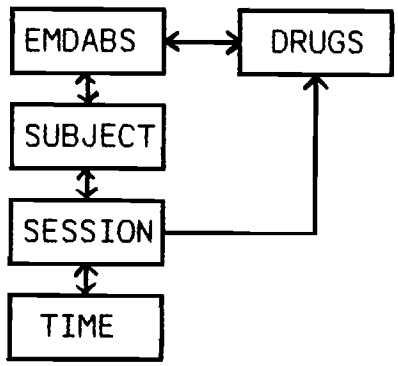
Wanneer alle componenten van Oracle geïnstalleerd worden, neemt dit op de harddisk 7.5 Mb in beslag. Het programma ORAINST.BAT verzorgt de gehele installatie (Lit. 5). Na deze (eenmalige) installatie is het mogelijk de in paragraaf 4.1 genoemde tools en utilities te gebruiken. Voorwaarde is wel dat het Oracle RDBMS aktief is. Dit aktiveren gebeurt door het invoeren van het commando ORACLE.

5 DE TWEEDE VERSIE VAN EMDABS

Na een beschrijving te hebben gegeven van de werking van het RDBMS Oracle volgt nu de beschrijving van de implementatie van EMDABS in dit systeem. Daarbij heb ik voornamelijk gebruik gemaakt van de applicatie generator SQL*FORMS. Uit het voorgaande hoofdstuk is al gebleken dat het werken met deze applicatie generator anders is dan het schrijven van een programma. Het is dan ook niet mogelijk de dBASE III programma's direkt te compileren naar Oracle programma's. Uitgaande van de in hoofdstuk 3 beschreven werking van EMDABS, ben ik gestart met een geheel nieuwe implementatie.

Zoals ook al in hoofdstuk 3 is vermeld, worden bij de eerste versie van EMDABS veel kleine database files (tabellen) dynamisch aangemaakt. Uit de volgende paragraaf zal blijken dat ik bij de nieuwe implementatie minder tabellen gebruik. De structuren van de gebruikte tabellen zijn dezelfde als die bij dBASE III gebruikt zijn (zie bijlage 1 en 2). De tabelnamen zijn wel anders, bijvoorbeeld EMDRUGDT.DBF heet nu DRUGDATA.

Overeenkomstig de figuren 3.1 en 3.2 heb ik gekozen EMDABS als volgt in forms onder te verdelen:



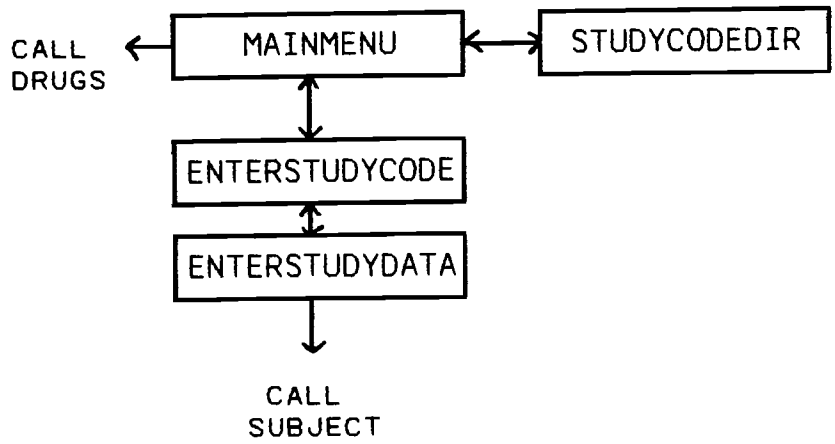
figuur 5.1. EMDABS in forms onderverdeeld.

Wegens tijdgebrek is het mij niet gelukt EMDABS volledig te implementeren. De forms EMDABS, SUBJECT en DRUGS zijn geheel afgerond. In de paragrafen 5.1 tot en met 5.3 zullen deze forms besproken worden. Alle karaktervelden van de genoemde forms hebben vanwege de eenduidigheid het attribuut "Uppercase" gekregen, zodat alle letters als hoofdletters ingevoerd worden.

5.1 De EMDABS form

De eerste form waarmee de applicatie EMDABS begint is genaamd EMDABS. EMDABS laat het hoofdmenu zien en geeft de

mogelijkheid om studiegegevens in te voeren (studie nivo). Deze form is opgebouwd uit vier blokken, welke verdeeld zijn over drie pagina's (schermen). In figuur 5.2 is schematisch weergegeven in welke volgorde de blokken doorlopen worden.



figuur 5.2. Blokstructuur van de EMDABS form.

De blokken ENTERSTUDYCODE en ENTERSTUDYDATA bevinden zich op dezelfde pagina. De werking van deze form komt overeen met de dBASE III programma's EMDABS en EMDAB1. Op form nivo heb ik een KEY-OTHERS trigger aangemaakt, welke alle funktietoetsen, zoals bijvoorbeeld [CREATE RECORD], [DELETE RECORD] en [EXIT], in de gehele form uitschakelt. Hierdoor wordt voorkomen dat de gebruiker (operator) ongewenste manipulaties kan uitvoeren. Deze trigger ziet er als volgt uit:

```
#EXEMACRO NULL;
```

Om nu toch bepaalde funktietoetsen te kunnen gebruiken, kan deze trigger "overruled" worden door een trigger op blok of veld nivo. In de volgende drie paragrafen worden de gebruikte blokken nader toegelicht.

5.1.1 Het EMDABS hoofdmenu

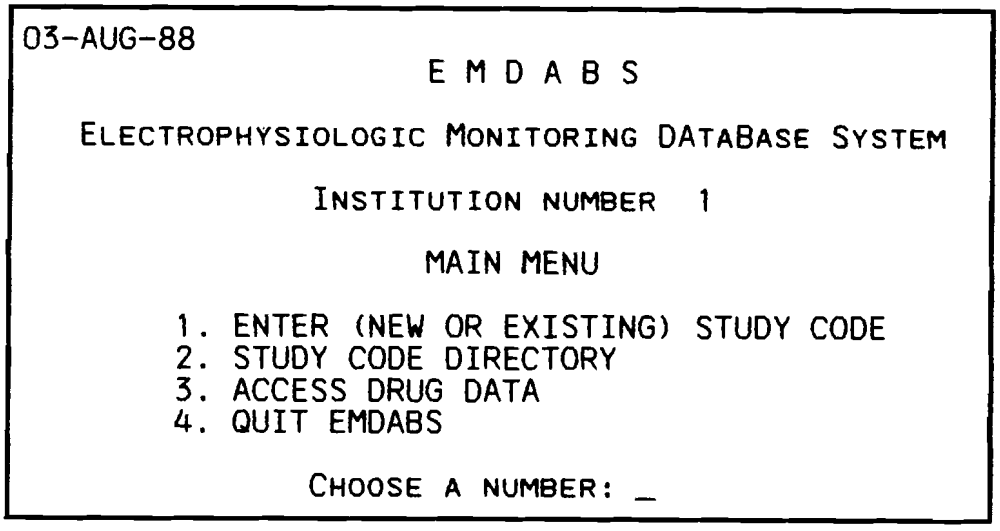
Mainmenu is het eerste blok en bevindt zich op de eerste pagina. Dit blok heeft geen basistabel, dat wil zeggen het blok is niet direkt verbonden met een tabel. Zo'n blok wordt een controle blok genoemd. De velden in een controle blok komen dus niet overeen met de velden in een tabel. Deze eerste pagina is weergegeven in figuur 5.3. De velden, die in dit blok gedefinieerd zijn, zijn SYSTEMDATE, INSTCODE en CHOICE1. Voordat het blok op het scherm verschijnt, worden de volgende PRE-BLOCK triggers uitgevoerd:

```
SELECT SYSDATE                  SELECT INSTCODE
INTO   MAINMENU.SYSTEMDATE      INTO   MAINMENU.INSTCODE
FROM   DUAL                     FROM   INSTITUTIONCODE
```

De eerste trigger zet de systeemdatum in het veld SYSTEMDATE. Omdat hierbij geen gegevens uit een tabel gehaald hoeven te worden, kan in principe elke bestaande tabelnaam achter de FROM clausule ingevuld worden. Om de select snel uit te voeren wordt daarvoor de tabel DUAL gebruikt. Dit is een tabel met één kolom en één rij, zodat de toegangstijd voor deze tabel minimaal is.

De tweede trigger zet het nummer van het instituut in het veld INSTCODE. Dit nummer staat in de tabel INSTITUTIONCODE. Wanneer EMDABS bij een ander instituut geïnstalleerd wordt, moet een ander nummer in de tabel gezet worden. Dit kan eenvoudig gebeuren met het tabelinstallatie programma, dat weergegeven is in bijlage 2. De velden SYSTEMDATE en INSTCODE zijn voorzien van een "no input" attribuut, zodat de gegevens tijdens een run niet veranderd kunnen worden. Bij het veld CHOICE1 kunnen wel gegevens ingevoerd worden. Op dit veld is een KEY-NXTFLD trigger gedefinieerd. Dit is een trigger die in aktie komt als op de ENTER toets wordt gedrukt. Deze trigger maakt gebruik van het SQL*FORMS commando CASE:

```
#EXEMACRO CASE CHOICE1 IS
WHEN '1'    THEN GOBLK ENTERSTUDYCODE;
WHEN '2'    THEN GOBLK STUDYCODEDIR;EXEQRY;
WHEN '3'    THEN CALL DRUGS;
WHEN '4'    THEN EXIT;
WHEN OTHERS THEN NULL;
END CASE;
```



figuur 5.3. Het EMDABS hoofdmenu.

Wanneer een 1 ingevoerd wordt, wordt verder gegaan met het blok ENTERSTUDYCODE. Wanneer 2 wordt ingevoerd wordt verder gegaan met het blok STUDYCODEDIR en wordt een query op

dit blok uitgevoerd (zie paragraaf 5.1.2). Het invoeren van een 3 heeft tot gevolg dat door middel van het CALL commando de form DRUGS aangeroepen wordt. EMDABS kan gestopt worden door een 4 in te voeren. Het invoeren van elk ander karakter heeft geen enkele aktie tot gevolg.

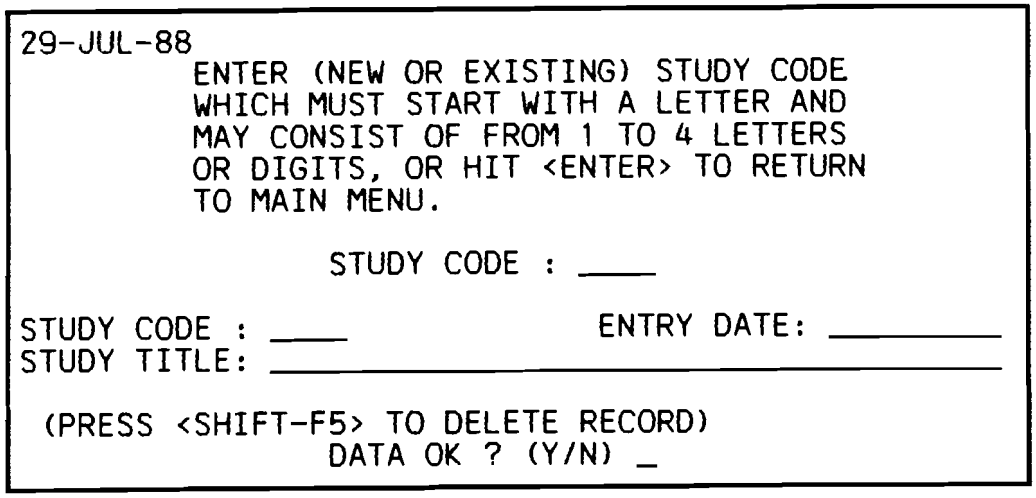
5.1.2 Het overzicht van de studiegegevens

Bij keuzemogelijkheid 2 van het hoofdmenu wordt naar de het blok STUDYCODEDIR op de tweede pagina overgegaan. Dit blok, met STUDYDATA als basistabel, geeft een overzicht van alle studie codes met bijbehorende datum en titel. De tabel STUDYDATA heeft dezelde structuur als EMSTUDY.DBF bij de dBASE III versie, alleen het veld ABREVCODE ontbreekt. Zoals al in hoofdstuk 3 gezegd is, is het bepalen van een afgekorte studie code, vanwege de grotere lengte van de tabelnamen, bij Oracle niet nodig. Het weergeven van meerdere rijen uit een tabel is niet mogelijk met met behulp van het SELECT INTO commando. Daarvoor wordt het EXEQRY (execute query) commando gebruikt. Wanneer men geen voorwaarde opgeeft, levert het uitvoeren van een query alle rijen uit de tabel op.

Omdat de studie titel 125 karakters groot kan zijn en op het scherm maar 62 karakters weergegeven worden, is het mogelijk met behulp van de pijltoetsen de studie titels te scrollen. Wanneer het aantal rijen in de tabel STUDYDATA groter is dan het aantal rijen op het scherm, kan ook met de pijltoetsen op en neer gescrold worden. Door op [ENTER] te drukken wordt weer teruggekeerd naar het hoofdmenu.

5.1.3 Het invoeren van de studiegegevens

Door in het hoofdmenu nummer 1 te kiezen kom je op de derde pagina terecht (zie figuur 5.4).



figuur 5.4. Het manipuleren van studiegegevens met de EMDABS form.

Op deze pagina bevinden zich twee blokken, namelijk het controle blok ENTERSTUDYCODE en het blok ENTERSTUDYDATA. Deze splitsing in twee blokken is gedaan om een query-voorwaarde in het blok ENTERSTUDYDATA te kunnen formuleren.

Het blok ENTERSTUDYCODE bestaat uit twee velden, namelijk SYSTEMDATE en STUDY_CODE. De datum in het eerstgenoemde veld is een kopie van de datum in het hoofdmenu. Dit kopieeren van gegevens kan gerealiseerd worden door in het (SQL*FORMS) DEFINE FIELD menu de VALIDATION aktie te selecteren, waarna in het SPECIFY VALIDATION menu het desbetreffende blok en veld opgegeven kunnen worden. Met behulp van hetzelfde menu is voor het veld STUDY_CODE een onder- en bovengrens van respectievelijk A000 en ZZZZ opgegeven, waardoor alleen studie codes geaccepteerd worden, die met een letter beginnen. De KEY-NXTFLD trigger die op het veld STUDY_CODE is gedefinieerd, ziet er als volgt uit:

```
#EXEMACRO CASE ENTERSTUDYCODE.STUDY_CODE IS
WHEN ''     THEN GOBLK MAINMENU;
WHEN OTHERS THEN EXETRG QUERYSTUDYDATA;
END CASE;
```

Deze trigger zorgt ervoor, dat wanneer niets ingevoerd wordt, teruggekeerd wordt naar het hoofdmenu, en in alle andere gevallen wordt de trigger met naam QUERYSTUDYDATA aangeroepen. Deze trigger bestaat uit drie stappen:

1. #COPY ENTERSTUDYCODE.STUDY_CODE GLOBAL.STCODE

Stap 1 kopieert de ingevoerde studie code naar de variabele GLOBAL.STCODE, welke in de SUBJECT form gebruikt wordt. Met behulp van het COPY commando kunnen dus variabelen tussen de verschillende forms uitgewisseld worden.

2. SELECT 'X'
   FROM STUDYDATA
   WHERE STUDYDATA.STUDY_CODE=:ENTERSTUDYCODE.STUDY_CODE

Stap 2 kijkt of de ingevoerde studie code al in de tabel STUDYDATA voorkomt. Deze stap is toegevoegd om de boodschap "New study code" te laten verschijnen, wanneer de studie code nog niet in de tabel voorkomt. Het is namelijk mogelijk een boodschap op het scherm te laten verschijnen, als een triggerstap mislukt. Omdat het hier alleen om de WHERE clausule gaat, wordt om snelheid te winnen niet een kolomnaam maar een constante 'X' geselecteerd.

3. #EXEMACRO NXTBLK;CLRBLK;EXEQRY;

Deze laatste stap gaat naar het volgende blok (ENTERSTUDYDATA) en wist de eventueel aanwezige informatie. Daarna wordt een

query op de bijbehorende basistabel STUDYDATA uitgevoerd. Doordat het veld STUDY_CODE in het blok ENTERSTUDYDATA een kopie is van het veld STUDY_CODE in het blok ENTERSTUDYCODE, betekent het uitvoeren van de query dus het weergeven van dat record, waarvoor geldt dat de studie code gelijk is aan de ingevoerde studie code.

Na deze trigger worden in het geval van een bestaande studie code dus ook de overige studiegegevens (datum en titel) weergegeven en bestaat de mogelijkheid deze te veranderen. In geval van een nieuwe code kunnen de studie titel en de datum ingevoerd worden. Wanneer men het gehele record uit de tabel wil verwijderen moet shift-F5 ingedrukt worden. Deze funktietoets is namelijk in dit blok door een trigger aktief gemaakt.

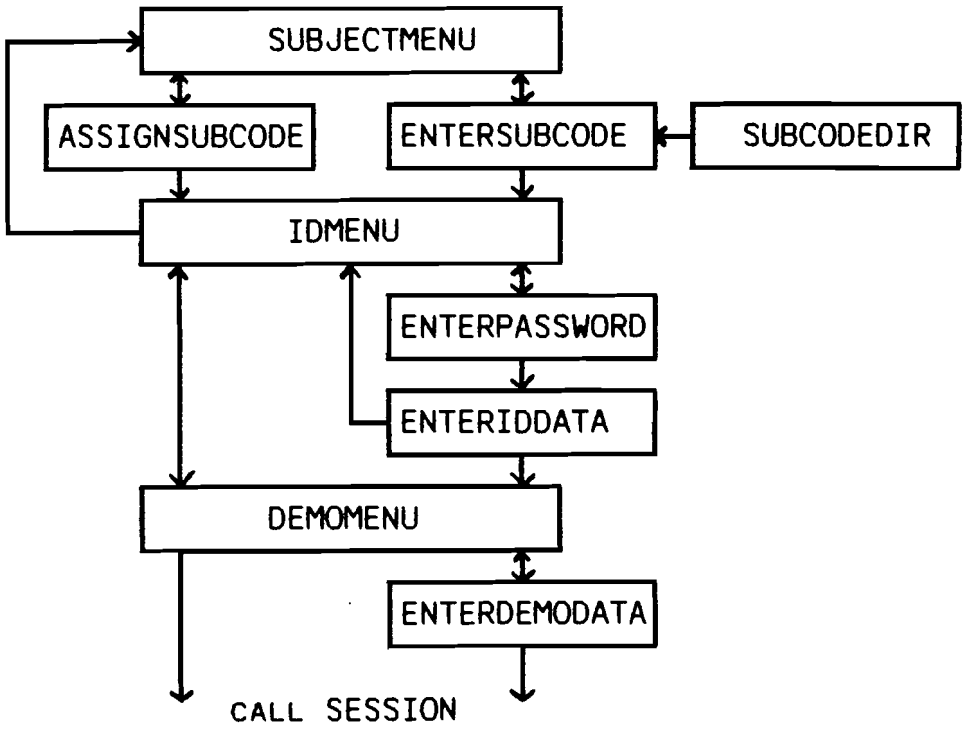
Als de studiegegevens ingevoerd of veranderd zijn, kan men tenslotte met behulp van het veld CHOICE2 een keuze maken om al dan niet verder te gaan. De KEY-NXTFLD trigger die dit organiseert ziet er als volgt uit:

```
#EXEMACRO CASE CHOICE2 IS
WHEN 'Y'    THEN COMMIT;GOBLK MAINMENU;CALL SUBJECT;
WHEN 'N'    THEN GOBLK ENTERSTUDYCODE;CLRBLK;
WHEN OTHERS THEN NULL;
END CASE;
```

Wanneer "Y" wordt ingevoerd, worden alle gegevens definitief in de tabel gezet (COMMIT) en wordt de SUBJECT form aangeroepen. In geval van "N" wordt naar het vorige blok teruggekeerd.

5.2 De SUBJECT form

Nadat met de vorige form de studiegegevens ingevoerd zijn, komen de subjectgegevens aan bod (subject nivo). De SUBJECT form komt overeen met de dBASE III programma's EMCODEMN, EMCODEM1, EMIDMENU en EMDEMOMN. Figuur 5.5 laat de verdeling in blokken zien. De blokken ENTERSUBCODE en SUBCODEDIR bevinden zich op dezelfde pagina. Dit geldt ook voor de blokken ENTERPASSWORD en ENTERIDDATA. Op form nivo is weer een KEY-OTHERS trigger gedefinieerd die alle funktie toetsen uitschakelt. Een tweede trigger van het type PRE-FORM kopieert de variabele GLOBAL.STCODE in het veld STUDY_CODE van het eerste blok (SUBJECTMENU).



figuur 5.5. Blokstructuur van de SUBJECT form.

5.2.1 Het subject code menu

Het eerste blok van de SUBJECT form is SUBJECTMENU. Hierbij wordt het volgende menu getoond:

1. NEW SUBJECT: ASSIGN SUBJECT CODE
2. ENTER EXISTING SUBJECT CODE
3. RETURN TO MAIN MENU

De keuze kan in het veld CHOICE1 ingevoerd worden. De KEY-NXTFLD trigger, die op dit veld gedefinieerd is, maakt weer gebruik van het CASE commando (zie paragraaf 5.1). Bij keuze 1 wordt naar het blok ASSIGNSUBCODE gegaan en wordt de trigger genaamd ASSIGN uitgevoerd. Bij keuze 2 wordt naar het blok ENTERSUBCODE gesprongen en vindt een query in het blok SUBCODEDIR plaats. Keuze 3 resulteert door middel van een EXIT in het terugkeren naar het EMDABS hoofdmenu.

5.2.2 Het automatisch toekennen van een subject code

Wanneer men aan een studie een nieuw subject wil toevoegen, moet aan dit subject een nieuwe code toegewezen worden. Dit wordt met behulp van het blok ASSIGNCODE en de bovengenoemde

trigger ASSIGN gerealiseerd. Deze trigger, op form nivo gedefinieerd, bestaat uit twee stappen:

1. SELECT 'X'
   FROM SESSIONDATA
   WHERE SESSIONDATA.STUDY_CODE=:SUBJECTMENU.STUDY_CODE

Deze eerste stap kijkt of er al een subject bij de huidige studie is opgenomen. Indien de studie code in de tabel SESSIONDATA voorkomt en er dus al eerder een subject is ingevoerd, slaagt de trigger en wordt verder gegaan met de tweede stap. In geval van een eerste subject mislukt de trigger en wordt afgebroken.

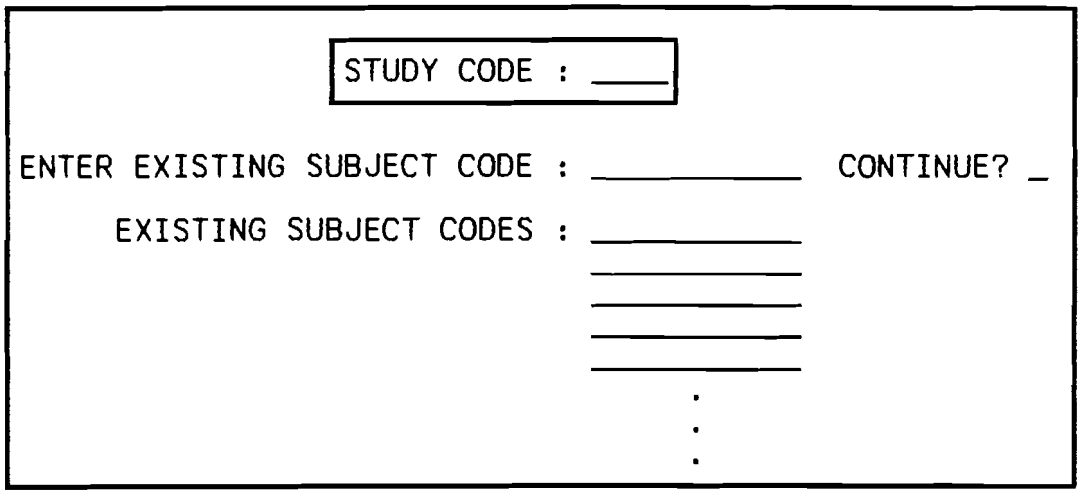
2. SELECT MAX(SUB_CODE)+1
   INTO ASSIGNSUBCODE.SUB_CODE
   FROM SESSIONDATA
   WHERE SESSIONDATA.STUDY_CODE=:SUBJECTMENU.STUDY_CODE

Deze stap bepaalt de grootste subject code in de tabel SESSIONDATA bij de huidige studie code en telt daar 1 bij op. Vervolgens wordt deze nieuwe subject code op het scherm in het veld SUB_CODE zichtbaar gemaakt. Omdat het veld SUB_CODE op het scherm, met behulp van het SPECIFY VALIDATION menu, de default waarde 10011110001 heeft gekregen, wordt in geval van een eerste subject bij een bepaalde studie dit de bijbehorende subject code. Nadat de code is toegewezen wordt gevraagd of er verder gegaan moet worden. De KEY-NXTFLD trigger, die op het veld CHOICE2 gedefinieerd is, zorgt ervoor, dat in geval van "Y" de studie en subject code in de tabel SESSIONDATA worden opgeslagen en dat naar het blok IDMENU gegaan wordt. In geval van "N" wordt teruggekeerd naar het subject code menu. De tweede stap van de trigger kopieert de gevonden subject code naar het blok IDMENU.

5.2.3 Het invoeren van een bestaande subject code

Het blok ENTERSUBCODE maakt het mogelijk een bestaande subject code in te voeren, waarbij het blok SUBCODEDIR op hetzelfde scherm een overzicht geeft van de in de tabel SESSIONDATA aanwezige subject codes. Het scherm is weergegeven in figuur 5.6. Bij keuze 2 van het subject menu vindt er op het blok SUBCODEDIR een query plaats. Doordat bij dit blok de onderstaande WHERE en ORDER BY clausules gedefinieerd zijn (met behulp van de ORDERING aktie in het DEFINE BLOCK menu), resulteert een query op dit blok in een geordende weergave van alleen die subject codes, die horen bij de eerder ingevoerde studie code.

```
WHERE SESSIONDATA.STUDY_CODE=:SUBJECTMENU.STUDY_CODE
ORDER BY SUB_CODE
```



figuur 5.6. Invoeren van een bestaande subject code in de SUBJECT form.

Nadat de bestaande subject codes weergegeven zijn kan een van deze codes ingevoerd worden. Een KEY-NXTFLD trigger controleert of de ingevoerde subject code inderdaad in de tabel SESSIONDATA voorkomt. Als dit niet het geval is, wordt teruggekeerd naar het subject code menu. Als dit wel het geval is kan men in het veld CHOICE3 aangeven of men al dan niet verder wil gaan. Bij de keuze "Y" zorgt een KEY-NXTFLD trigger ervoor dat de ingevoerde subject code samen met de studie code in de tabel SESSIONDATA worden opgeslagen. Bij de keuze "N" wordt weer teruggekeerd naar het subject code menu. De tweede stap van deze trigger kopieert de ingevoerde subject code naar het blok IDMENU.

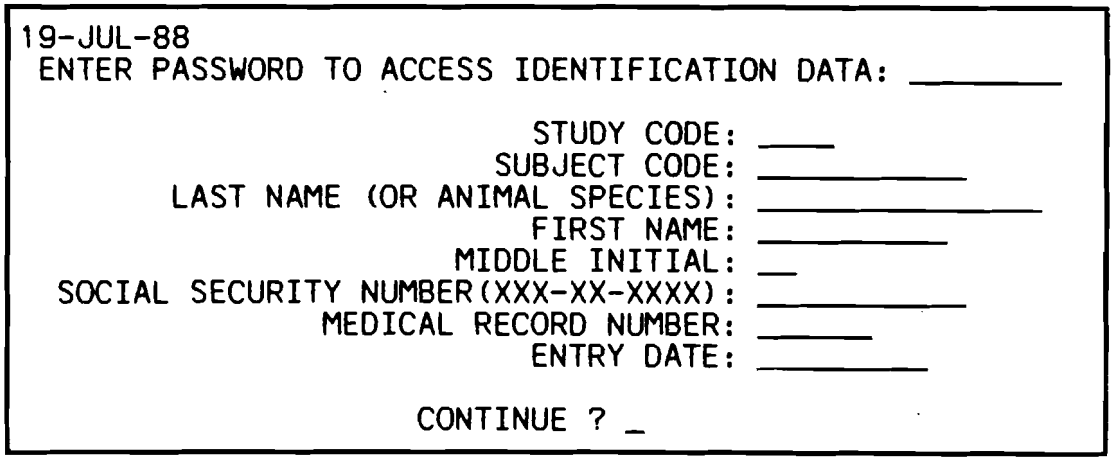
5.2.4 Het invoeren van identificatie gegevens

Nu de studie en subject codes bekend zijn, kan worden verder gegaan met de identificatie gegevens. Het eerste blok dat zich hiermee bezig houdt is IDMENU, waarbij het volgende menu verschijnt:

1. ENTER/EDIT/REVIEW IDENTIFICATION DATA
2. RETURN TO SUBJECT MENU
3. BYPASS IDENTIFICATION BLOCK

Het gekozen nummer wordt in het veld CHOICE4 ingevoerd. De KEY-NXTFLD trigger die bij dit veld hoort zorgt ervoor dat bij keuze 1 verder gegaan wordt met het blok ENTERPASSWORD. Bij keuze 2 wordt naar het subject code menu teruggekeerd en bij keuze 3 wordt overgegaan naar het blok DEMOMENU.

De blokken ENTERPASSWORD en ENTERIDDATA bevinden zich op dezelfde pagina. Deze pagina ziet er als volgt uit:



figuur 5.7 Manipuleren van identificatie gegevens met de SUBJECT form.

Voordat de identificatie gegevens in het blok ENTERIDDATA gemanipuleerd kunnen worden, moet eerst in het blok ENTERPASSWORD een toegangscode in het veld SECRET ingevoerd worden. Bij een juiste toegangscode wordt door middel van een EXEQRY de bij de huidige studie en subject code behorende identificatie gegevens uit de tabel IDDATA zichtbaar gemaakt, waarna ze veranderd of ingevoerd kunnen worden. Tenslotte kan gekozen worden al of niet door te gaan. De KEY-NXTFLD trigger die op het veld CHOICE5 gedefinieerd is, gaat bij "Y" verder met het blok DEMOMENU en bij "N" terug naar het identificatie gegevens menu. In het blok ENTERIDDATA is met behulp van een KEY-DELREC trigger de [DELETE RECORD] funktie toets aktief gemaakt, zodat door indrukken van shift-F5 het gehele record uit de tabel IDDATA verwijderd wordt.

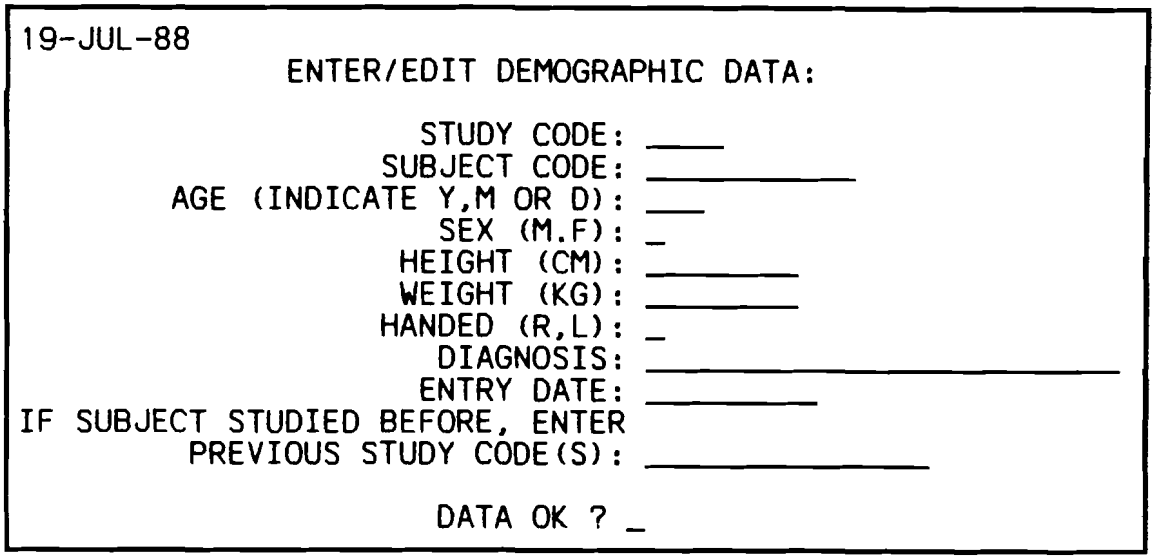
5.2.5 Het invoeren van demografische gegevens

Voordat de demografische gegevens in het blok ENTERDEMODATA ingevoerd of veranderd kunnen worden, laat het blok DEMOMENU het volgende menu zien:

1. ENTER/EDIT/REVIEW DEMOGRAPHIC DATA
2. RETURN TO IDENTIFICATION DATA MENU
3. BYPASS DEMOGRAPHIC BLOCK

De keuze wordt ingevoerd in het veld CHOICE6. De bijbehorende KEY-NXTFLD trigger zorgt ervoor dat bij keuze 1 naar het blok ENTERDEMODATA wordt gegaan, waarna een query op de bij dit blok behorende tabel DEMOGRDATA wordt uitgevoerd. Bij keuze 2 wordt teruggekeerd naar het identificatie gegevens menu en keuze 3 resulteert in het aanroepen van de SESSION form.

Indien 1 gekozen is, wordt het ENTERDEMODATA blok zichtbaar:

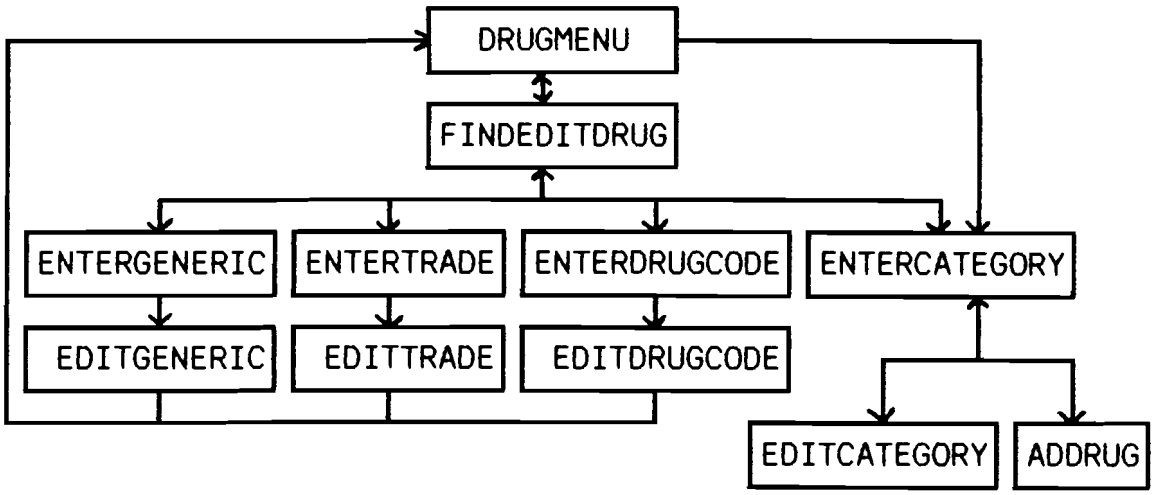


figuur 5.8. Manipuleren van demografische gegevens met behulp van de SUBJECT form.

De query die hierop volgt, laat de demografische gegevens uit de tabel DEMOGRDATA zien die bij de huidige studie en subject horen. Dan kan de gebruiker deze gegevens veranderen of kan nieuwe gegevens toevoegen. Met behulp van de KEY-DELREC trigger kan in dit blok het gehele record uit de tabel DEMOGRDATA ook verwijderd worden. In het veld CHOICE7 kan weer gekozen worden wel of niet verder te gaan. In de eerste stap van de bijbehorende KEY-NXTFLD trigger wordt de subject code naar de variabele GLOBAL.SUBCODE gekopieerd, zodat deze code, net als de studie code, aan de SESSION form overgedragen kan worden. De tweede stap houdt zich bezig met de ingevoerde keuzemogelijkheid. Bij "Y" wordt de SESSION form aangeroepen en bij "N" wordt naar het demografische gegevens menu teruggekeerd.

5.3 De DRUGS form

Deze form stelt de gebruiker in staat om gegevens met betrekking tot medicijnen (code, generieke naam en handelsnamen) op te zoeken, te veranderen of te verwijderen. Deze gegevens zijn opgeslagen in de tabel DRUGDATA, welke dezelfde structuur heeft als de dBASE III database file EMDRUGDT.DBF. De DRUGS form is als volgt in blokken verdeeld:



figuur 5.9. Blokstructuur van de DRUGS form.

Door middel van een KEY-OTHERS trigger zijn de funktietoetsen weer uitgeschakeld. De blokken ENTERGENERIC en EDITGENERIC bevinden zich op één pagina. Dit geldt ook voor ENTERTRADE, EDITTRADE en ENTERDRUGCODE, EDITDRUGCODE. De werking van deze form komt overeen met de dBASE III programma's EMDRUGMN, EMFINDRG, EMEDITDRG en EMADDRUG. In de paragrafen 5.3.1 tot en met 5.3.3 worden de blokken nader toegelicht.

5.3.1 Het drug menu

Het eerste (controle) blok is genaamd DRUGMENU en laat het volgende menu zien:

1. FIND/EDIT DRUG CODE OR NAME(S)
2. ADD/DELETE DRUG DATA
3. RETURN TO MAIN MENU

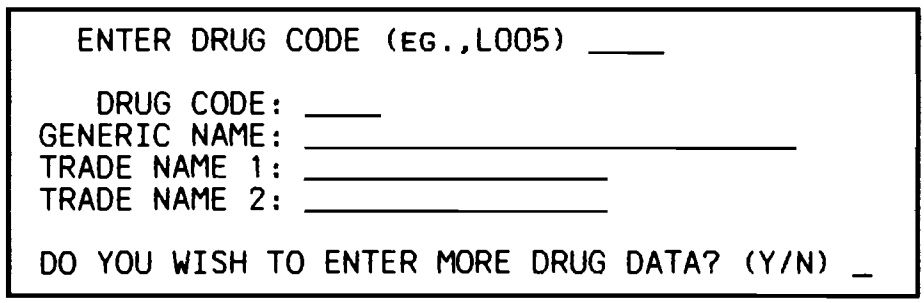
De KEY-NXTFLD trigger van het veld CHOICE1 zorgt ervoor dat bij keuze 1 wordt verder gegaan met het blok FINDEDITDRUG. Bij keuze 2 wordt overgegaan naar het blok ENTERCATEGORY en bij keuze 3 wordt naar het EMDABS hoofdmenu teruggekeerd.

5.3.2 Het opzoeken van drugs

Het opzoeken van gegevens begint met het blok FINDEDITDRUG en geeft het volgende menu:

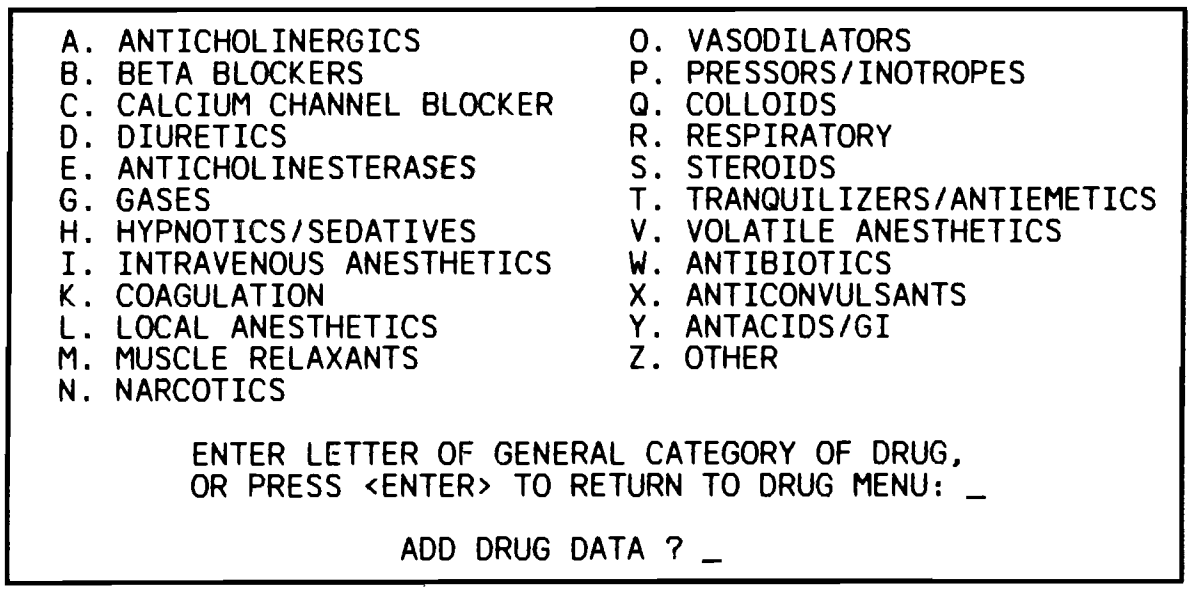
1. ENTER GENERIC NAME OF DRUG TO FIND/EDIT DRUG DATA
2. ENTER TRADE NAME OF DRUG TO FIND/EDIT DRUG DATA
3. ENTER DRUG CODE TO FIND/EDIT DRUG DATA
4. ENTER GENERAL CATEGORY OF DRUG TO FIND/EDIT DRUG DATA
5. RETURN TO DRUG MENU

De KEY-NXTFLD trigger van het veld CHOICE2 zorgt ervoor dat bij keuzemogelijkheden 1,2,3 en 4 wordt overgegaan naar respectievelijk ENTERGENERIC, ENTERTRADE, ENTERDRUGCODE en ENTERCATEGORY. Bij keuze 5 wordt weer teruggekeerd naar het vorige menu. Bij de keuzemogelijkheden 1,2 en 3 gebeurt steeds hetzelde, namelijk een bekend gegeven van een medicijn wordt in een controle blok (ENTERGENERIC, ENTERTRADE, ENTERDRUGCODE) ingevoerd, waarna dit gegeven naar het volgende blok wordt gekopieerd en door middel van een query de bijbehorende gegevens zichtbaar worden gemaakt. De getoonde gegevens kunnen ook nog veranderd worden. De indeling van het scherm ziet er bijvoorbeeld bij de blokken ENTERDRUGCODE en EDITDRUGCODE als volgt uit:



figuur 5.10. Opzoeken en veranderen van drugdata.

Als nog meer gegevens ingevoerd dienen te worden, wordt weer naar het drug menu gegaan. Als dit niet het geval is, wordt naar het EMDABS hoofdmenu gegaan. Bij keuze 4 (en bij keuze 1 uit het drug menu) verschijnt het blok ENTERCATEGORY:

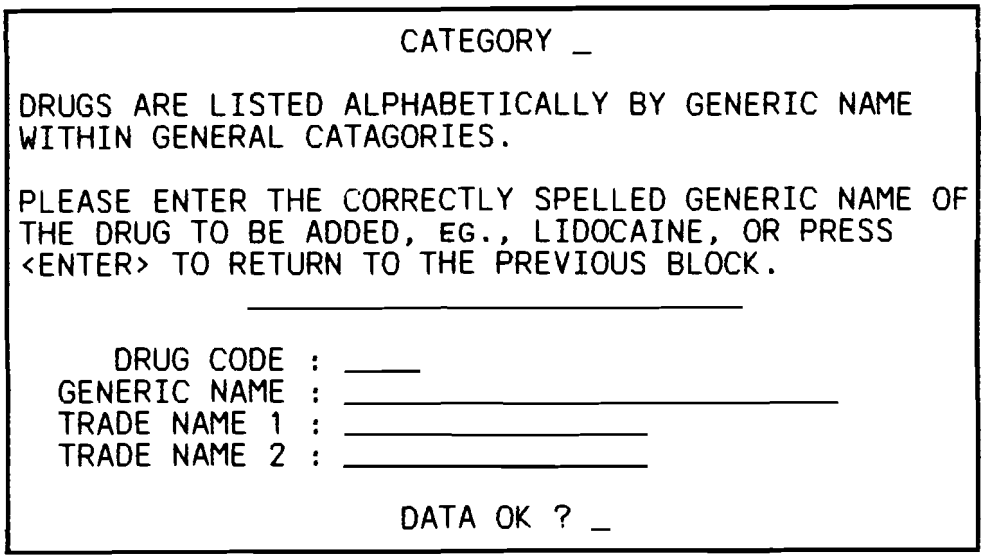


figuur 5.11. Het kiezen van een kategorie.

Nadat de gewenste kategorie is ingevoerd en men geen nieuwe medicijnen wil toevoegen, worden in het blok EDITCATEGORY al de bij deze kategorie behorende gegevens uit de tabel DRUGDATA getoond en bestaat de mogelijkheid gegevens te veranderen of te verwijderen. Daartoe zijn een aantal funktietoetsen ([DELREC], [NXTSET], [END] en [CANCEL]) met behulp van triggers geaktiveerd.

5.3.3 Het toevoegen van nieuwe drugs

Wanneer men in het blok ENTERCATEGORY kiest om bij de gekozen kategorie een nieuw medicijn toe te voegen, wordt het blok ADDRUG zichtbaar:



figuur 5.12. Het toevoegen van nieuwe medicijnen.

Nadat de generieke naam is ingevoerd wordt door middel van een KEY-NXTFLD trigger een nieuwe drug code aan dit medicijn toegekend. Dit gebeurt door de grootste drug code binnen die kategorie te zoeken en daar 1 bij op te tellen. Een voorbeeld (zie ook hoofdstuk 3): Stel we willen halothaan toevoegen aan de gassen lucht en helium. Het resultaat komt er dan als volgt uit te zien:

```
G001 AIR
G003 HALOTHANE
G002 HELIUM
```

Als de drug code bekend is, is het nog mogelijk een of twee handelsnamen toe te voegen. Als de gegevens goed zijn ingevoerd wordt naar het blok ENTERCATEGORY teruggekeerd.

5.4 Vergelijken van de oude en de nieuwe versie

Wanneer we de dBASE III versie van EMDABS met de Oracle versie vergelijken, zien we dat er naar de gebruiker toe weinig is veranderd: De menu's zijn grotendeels hetzelfde gebleven en ook de volgorde waarin ze doorlopen worden. De schermindeling is echter anders geworden.

De nieuwe versie gebruikt tot zover minder tabellen. Dit is bereikt door aan de tabellen SESSIONDATA, IDDATA en DEMOGRDATA een kolom STUDY_CODE toe te voegen. In de dBASE versie werd uit oogpunt van de privacy de studie code in de database filenaam verwerkt. Zoals al in paragraaf 4.2.4 is vermeld, bezit Oracle een groot aantal hulpmiddelen om de privacy binnen de Oracle database te garanderen.

Wat betreft de subject en drug codes zijn is er ook wat veranderd. Zo wordt de subject code nu zonder streepjes ingevoerd, dus bijvoorbeeld 10011110001 in plaats van 1001-111-0001. De medicijnen worden nu op een andere manier gecodeerd. Zoals in paragraaf 3.3 opgemerkt is, kan bij de oude manier van coderen slechts een beperkt aantal medicijnen worden toegevoegd. Vandaar dat ik gekozen heb voor de in de vorige paragraaf besproken codering.

5.5 Het installeren van EMDABS

Als EMDABS op een nieuw systeem geïnstalleerd moet worden, moeten de volgende handelingen eenmalig worden uitgevoerd:

1. Installeren (en aktiveren) van Oracle (zie paragraaf 4.7).
2. Aanmaken van een gebruikerscode en toegangscode (zie paragraaf 4.2.4).
3. SQL*PLUS starten en de tabellen installeren met het tabel-installatieprogramma tablinst.sql (zie bijlage 2).
4. De programma's EMDABS.FRM, SUBJECT.FRM en DRUGS.FRM naar de harde schijf kopiëren.

Wanneer de forms op het nieuwe systeem ook veranderd moeten worden, moet de volgende stap toegevoegd worden:

5. SQL*FORMS starten en met behulp van de LOAD aktie de files EMDABS.INP, SUBJECT.INP en DRUGS.INP inladen.

EMDABS kan nu gestart worden met behulp van RUNFORM:

```
RUNFORM EMDABS gebruikerscode/toegangscode
```

6 CONCLUSIES

Het ontwerpen van een applicatie met behulp van de menugestuurde applicatie generator SQL*FORMS werkt goed: Met de screen painter kan de schermindeling gemakkelijk worden bepaald en de triggers maken het mogelijk om met eenvoudige commando's de gegevens te manipuleren.

Het implementeren van EMDABS in Oracle heb ik wegens tijdgebrek niet als geheel kunnen afronden. De gerealiseerde forms EMDABS, SUBJECT en DRUGS blijken aan de gestelde eisen te voldoen: De gebruikersvriendelijke, menugestuurde invoer zorgt voor een standaardisatie van de gegevens, wat noodzakelijk is voor de uitwisseling van deze gegevens tussen de verschillende instituten en specialisten.
Doordat Oracle voor praktisch alle computersystemen beschikbaar is en de gegevens tussen deze systemen uitwisselbaar zijn, is ook aan de eis voldaan dat EMDABS geschikt moet zijn voor verschillende computer hardware.

Voor zover EMDABS in Oracle geïmplementeerd is, kan geconcludeerd worden dat Oracle geschikt is voor het manipuleren van de verschillende soorten gegevens. Omdat ik niet met grote gegevensbestanden gewerkt heb, kan ik niets zeggen over de snelheid van Oracle in vergelijking tot dBASE III.

Nog te verrichten werkzaamheden / aanbevelingen:

- De forms SESSION en TIME moeten nog met SQL*FORMS geïmplementeerd worden.
- De gebruikersvriendelijkheid kan verhoogd worden door helpteksten op het scherm te laten verschijnen als de gebruiker hierom vraagt. Hiertoe zijn binnen SQL*FORMS mogelijkheden beschikbaar.
- Tijdgegevens kunnen in een later stadium met behulp van SQL*GRAPH grafisch weergegeven worden.

LITERATUUROVERZICHT

1. Eilers, H.B., Jansen, W.F., Volder, H.H.J. de : SQL in de praktijk, Academic service, Den Haag, 1986.

2. Fife, D.W., Hardgrave, W.T., Deutsch, D.R.: Database concepts, South-western publishing Co., Cincinnati, 1986.

3. Mayne, A., Wood, M.B.: Inleiding relationele databases, Samsom, Alphen a/d Rijn, 1985.

4. Oracle database administrator's guide, Oracle Corp., Belmont, California, USA, 1987.

5. Oracle installation and user's guide for IBM PC/MS DOS, Oracle Corp., Belmont, California, USA, 1987.

6. Oracle utilities user's guide, Oracle Corp., Belmont, California, USA, 1987.

7. SQL*FORMS designer's reference, Oracle Corp., Belmont, California, USA, 1987.

8. SQL*FORMS designer's tutorial, Oracle Corp., Belmont, California, USA, 1987.

9. SQL*PLUS reference guide, Oracle Corp., Belmont, California, USA, 1987.

10. SQL*PLUS user's guide, Oracle Corp., Belmont, California, USA, 1987.

11. Ter Bekke, J.H.: Database ontwerp, Stenfert Kroese, Leiden, 1983.

12. Theisen, G.J., Cluitmans, P.J.M., Beneken, J.E.W., Conlon, M., Grundy, B.L.: EMDABS, a multi-institutional research database system for electrofysiologic monitoring of the nervous system, Medinfo 86, Elsevier, 1986.

BIJLAGE 1. Definities van de tabellen die de dBASE III versie van EMDABS gebruikt.

Structuur van: EMSTUDY.dbf

Field	Field Name	Type	Width	Dec
1	STUDY_CODE	Character	4	
2	STUDYTITLE	Character	125	
3	INSTCODE	Character	2	
4	ENTRYDATE	Date	8	
5	ABREVCODE	Character	2	

Structuur van: EMSESSDT.dbf

Field	Field Name	Type	Width	Dec
1	STUDY_CODE	Character	4	
2	SUB_CODE	Character	13	
3	SESSION_N	Character	3	
4	SESS_DATE	Date	8	
5	ECGDBF	Character	1	
6	APDBF	Character	1	
7	PA_CVPDBF	Character	1	
8	VENTDBF	Character	1	
9	GASDBF	Character	1	
10	SATDBF	Character	1	
11	TEMPDBF	Character	1	
12	EEGDBF	Character	1	
13	EPDBF	Character	1	
14	EVENTDBF	Character	1	
15	OPERATION	Character	40	
16	ANESTHISTS	Character	35	
17	SURGEONS	Character	40	
18	TECHNOLOGY	Character	20	

Structuur van: EMDEMODT.dbf

Field	Field Name	Type	Width	Dec
1	SUB_CODE	Character	13	
2	AGE_YMD	Character	3	
3	SEX_MF	Character	1	
4	HEIGHT_CM	Numeric	6	2
5	WEIGHT_KG	Numeric	6	2
6	HANDED_RL	Character	1	
7	DIAGNOSIS	Character	40	
8	ENTRYDATE	Date	8	
9	PREVSTUDY	Character	15	
10	MEMO	Memo	10	

Structuur van: EMIDDATA.dbf

Field	Field Name	Type	Width	Dec
1	SUB_CODE	Character	13	
2	LAST_NAME	Character	15	
3	FIRST_NAME	Character	10	
4	MIDDLE_INT	Character	1	
5	SS_NUMBER	Character	11	
6	MED_RECORD	Character	6	
7	ENTRYDATE	Date	8	

Structuur van: EMDRUGDT.dbf

Field	Field Name	Type	Width	Dec
1	DRUG_CODE	Character	4	
2	GENERIC_N	Character	26	
3	TRADE_N1	Character	16	
4	TRADE_N2	Character	14	

Structuur van: EMEVNTSS.dbf

Field	Field Name	Type	Width	Dec
1	SUB_CODE	Character	13	
2	SESSION_N	Character	3	
3	SESS_SITE	Character	10	
4	IV_TYP_SIT	Character	30	
5	DRGPMPTYPE	Character	20	

Structuur van: EMEPSESS.dbf

Field	Field Name	Type	Width	Dec
1	SUB_CODE	Character	13	
2	SESSION_N	Character	2	
3	MON_TYPE	Character	15	
4	EP_TYPE	Character	10	
5	ELEC_TYPE	Character	10	
6	PLACE_TECH	Character	10	
7	CHANNELS	Character	3	
8	GRND_ELEC	Character	4	
9	MONTAGE	Character	50	
10	AMP_X_PHYS	Character	7	
11	FILTERS_HZ	Character	10	
12	SEN_MICROV	Character	3	
13	ANALY_MSEC	Character	5	
14	DATA_PTS	Character	4	
15	TMP_RSL_MS	Character	4	
16	SMPL_RT_HZ	Character	6	
17	STIM_TYPE	Character	15	
18	PRES_MODE	Character	15	
19	TRANSDUCER	Character	15	
20	STM_DUR_MS	Character	3	
21	STM_RT_HZ	Character	4	
22	STM_REP	Character	4	
23	STM_INT	Character	10	

Structuur van: EMEEGSES.dbf

Field	Field Name	Type	Width	Dec
1	SUB_CODE	Character	13	
2	SESSION_N	Character	2	
3	MON_TYPE	Character	15	
4	ELEC_TYPE	Character	10	
5	PLACE_TECH	Character	10	
6	CHANNELS	Character	3	
7	GRND_ELEC	Character	4	
8	MONTAGE	Character	100	
9	FILTERS_HZ	Character	10	
10	SEN_MICROV	Character	3	

Structuur van: EMECGSES.dbf

Field	Field Name	Type	Width	Dec
1	SUB_CODE	Character	13	
2	SESSION_N	Character	2	
3	MON_TYPE	Character	15	
4	LEAD_MONT	Character	10	
5	SAMPL_RATE	Character	10	

Structuur van: EMBPSESS.dbf

Field	Field Name	Type	Width	Dec
1	SUB_CODE	Character	13	
2	SESSION_N	Character	2	
3	MON_TYPE	Character	15	
4	INV_NONINV	Character	1	
5	MON_SITE	Character	10	
6	MON_METH	Character	10	
7	SAMPL_RATE	Character	10	

Structuur van: EMPASESS.dbf

Field	Field Name	Type	Width	Dec
1	SUB_CODE	Character	13	
2	SESSION_N	Character	2	
3	MON_TYPE	Character	15	
4	CATH_TYPE	Character	15	
5	PLAC_SITE	Character	20	
6	CO_MON	Character	15	
7	CO_TECHN	Character	10	
8	SAMPL_RATE	Character	10	

Structuur van: EMGASSES.dbf

Field	Field Name	Type	Width	Dec
1	SUB_CODE	Character	13	
2	SESSION_N	Character	2	
3	MON_TYPE	Character	15	
4	GAS_SAMPLD	Character	30	
5	SAMPL_SITE	Character	10	
6	SAMPL_RATE	Character	10	

Structuur van: EMVENTSS.dbf

Field	Field Name	Type	Width	Dec
1	SUB_CODE	Character	13	
2	SESSION_N	Character	2	
3	MON_TYPE	Character	15	
4	VENT_TYPE	Character	15	
5	CIRC_TYPE	Character	10	
6	SAMPL_RATE	Character	10	

Structuur van: EMSATSES.dbf

Field	Field Name	Type	Width	Dec
1	SUB_CODE	Character	13	
2	SESSION_N	Character	2	
3	MON_TYPE	Character	15	
4	PRB_LOC	Character	10	
5	SAMPL_RATE	Character	10	

Structuur van: EMTEMPSS.dbf

Field	Field Name	Type	Width	Dec
1	SUB_CODE	Character	13	
2	SESSION_N	Character	2	
3	MON_TYPE	Character	15	
4	PRB_LOC	Character	10	
5	SAMPL_RATE	Character	10	

Structuur van: EMEVNDAT.dbf

Field	Field Name	Type	Width	Dec
1	LTIME	Numeric	9	
2	TIME	Character	8	
3	EVN_CODE	Numeric	2	
4	EVN_TXT	Character	40	

Structuur van: EMEEGDAT.dbf

Field	Field Name	Type	Width	Dec
1	TIMTEXT	Character	8	
2	LTIM	Numeric	6	
3	MEANRE_STA	Numeric	3	
4	MEANRE_VAL	Numeric	4	1
5	ADINDL_STA	Numeric	3	
6	ADINDL_VAL	Numeric	6	3

Structuur van: EMPRSECG.dbf

Field	Field Name	Type	Width	Dec
1	TIME	Character	8	
2	LTIME	Numeric	6	
3	PSYS_STA	Numeric	3	
4	PSYS_VAL	Numeric	4	
5	MAP_STA	Numeric	3	
6	MAP_VAL	Numeric	4	
7	PDIA_STA	Numeric	3	
8	PDIA_VAL	Numeric	4	
9	HR_STA	Numeric	3	

10	HR_VAL	Numeric	4
11	DPAP_STA	Numeric	3
12	DPAP_VAL	Numeric	4
13	MPAP_STA	Numeric	3
14	MPAP_VAL	Numeric	4
15	MLAP_STA	Numeric	3
16	MLAP_VAL	Numeric	4
17	MCVP_STA	Numeric	3
18	MCVP_VAL	Numeric	4

Structuur van: EMGASDAT.dbf

Field	Field Name	Type	Width	Dec
1	TIME	Character	8	
2	CO2EX_STA	Numeric	3	
3	CO2EX_VAL	Numeric	4	
4	CO2IN_STA	Numeric	3	
5	CO2IN_VAL	Numeric	4	
6	O2EX_STA	Numeric	3	
7	O2EX_VAL	Numeric	4	
8	O2IN_STA	Numeric	3	
9	O2IN_VAL	Numeric	4	
10	N2EX_STA	Numeric	3	
11	N2EX_VAL	Numeric	4	
12	N2IN_STA	Numeric	3	
13	N2IN_VAL	Numeric	4	
14	N2OEX_STA	Numeric	3	
15	N2OEX_VAL	Numeric	4	
16	N2OIN_STA	Numeric	3	
17	N2OIN_VAL	Numeric	4	
18	ENFEX_STA	Numeric	3	
19	ENFEX_VAL	Numeric	4	
20	ENFIN_STA	Numeric	3	
21	ENFIN_VALI	Numeric	4	
22	HALEX_STA	Numeric	3	
23	HALEX_VAL	Numeric	4	
24	HALIN_STA	Numeric	3	
25	HALIN_VAL	Numeric	4	
26	ISOEX_STA	Numeric	3	
27	ISOEX_VAL	Numeric	4	
28	ISOIN_STA	Numeric	3	
29	ISOIN_VAL	Numeric	4	
30	RRATE_STA	Numeric	3	
31	RRATE_VAL	Numeric	4	
32	TIVOL_STA	Numeric	3	
33	TIVOL_VAL	Numeric	4	
34	PIP_STA	Numeric	3	
35	PIP_VAL	Numeric	4	
36	SAT_STA	Numeric	3	
37	SAT_VAL	Numeric	4	

Structuur van: EMTEMPDT.dbf

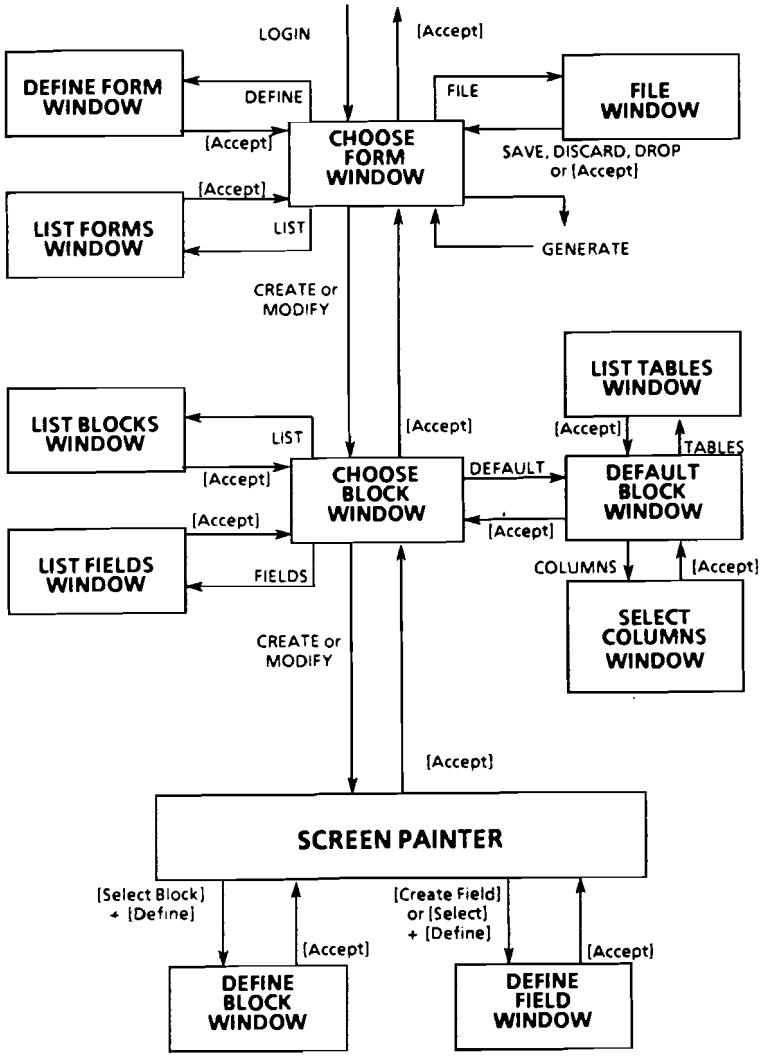
Field	Field Name	Type	Width	Dec
1	TIME	Character	8	
2	TCORE_STA	Numeric	3	
3	TCORE_VAL	Numeric	4	1
4	TSKIN_STA	Numeric	3	
5	TSKIN_VAL	Numeric	4	1
6	TORAL_STA	Numeric	3	
7	TORAL_VAL	Numeric	4	1
8	TNP_STA	Numeric	3	
9	TNP_VAL	Numeric	4	1
10	TESO_STA	Numeric	3	
11	TESO_VAL	Numeric	4	1
12	TREC_STA	Numeric	3	
13	TREC_VAL	Numeric	4	1

BIJLAGE 2. Het programma TABLINST.SQL installeert de tabel-definities (met eventueel de bijbehorende indexen), die door EMDABS gebruikt worden.

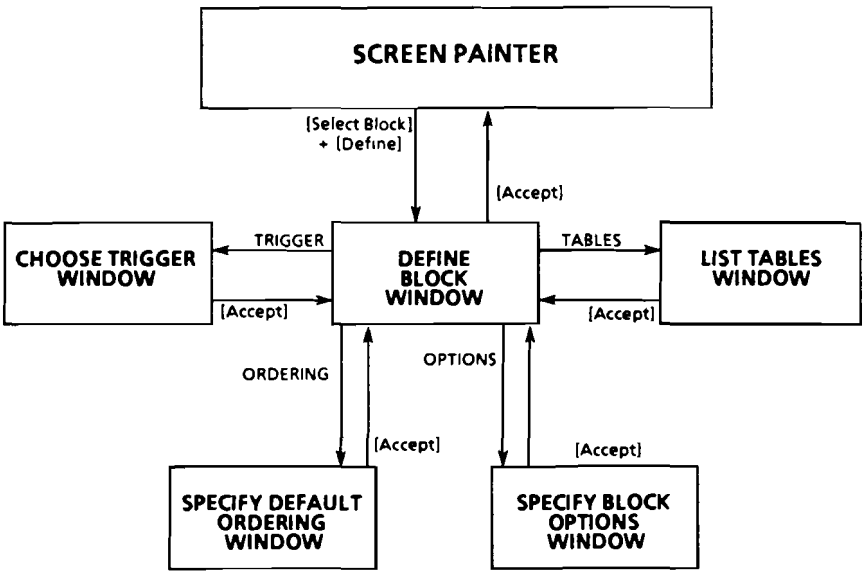
```
REM ************************************************************
REM * THIS PROGRAM INSTALLS THE TABLES AND INDICES USED BY EMDABS.*
REM * TYPE SQLPLUS USERCODE/PASSWORD AND @TABLINST TO RUN THIS    *
REM * INSTALLATION PROGRAM.                                       *
REM *                                                            *
REM *                          G.J. van Herwijnen, june 1988     *
REM ************************************************************

CREATE TABLE STUDYDATA
(STUDY_CODE CHAR(4) NOT NULL,
INSTCODE    NUMBER(2) NOT NULL,
ENTRYDATE   DATE,
STUDYTITLE  CHAR(125) NULL);

CREATE UNIQUE INDEX STUDYINDEX ON STUDYDATA(STUDY_CODE);

CREATE TABLE SESSIONDATA
(STUDY_CODE CHAR(4) NOT NULL,
SUB_CODE    NUMBER(11) NOT NULL,
SESSION_N   CHAR(3),
SESS_DATE   DATE,
ECGDBT      CHAR(1),
APDBT       CHAR(1),
PA_CVPDBT   CHAR(1),
VENTDBT     CHAR(1),
GASDBT      CHAR(1),
SATDBT      CHAR(1),
TEMPDBT     CHAR(1),
EEGDBT      CHAR(1),
EPDDBT      CHAR(1),
EVENTDBT    CHAR(1),
OPERATION   CHAR(40),
ANESTHISTS  CHAR(35),
SURGEONS    CHAR(40),
TECHNOLOGY  CHAR(20));

CREATE TABLE DRUGDATA
(DRUG_CODE CHAR(4) NOT NULL,
GENERIC_N  CHAR(26),
TRADE_N1   CHAR(16),
TRADE_N2   CHAR(14));

CREATE UNIQUE INDEX DRUGCODEINDEX ON DRUGDATA(DRUG_CODE);
```

```
CREATE TABLE IDDATA
(STUDY_CODE CHAR(4) NOT NULL,
SUB_CODE   NUMBER(11) NOT NULL,
ENTRYDATE  DATE,
LAST_NAME  CHAR(15),
FIRST_NAME CHAR(10),
MIDDLE_INT CHAR(2),
SS_NUMBER  CHAR(11),
MED_RECORD CHAR(6));

CREATE TABLE DEMOGRDATA
(STUDY_CODE CHAR(4) NOT NULL,
SUB_CODE  NUMBER(11) NOT NULL,
AGE_YMD   CHAR(3),
SEX_MF    CHAR(1),
HEIGHT_CM NUMBER(6,2),
WEIGHT_KG NUMBER(6,2),
HANDED_RL CHAR(1),
DIAGNOSIS CHAR(40),
ENTRYDATE DATE,
PREVSTUDY CHAR(15));

CREATE TABLE INSTITUTIONCODE
(INSTCODE NUMBER(2) NOT NULL);

INSERT INTO INSTITUTIONCODE VALUES (1);
```

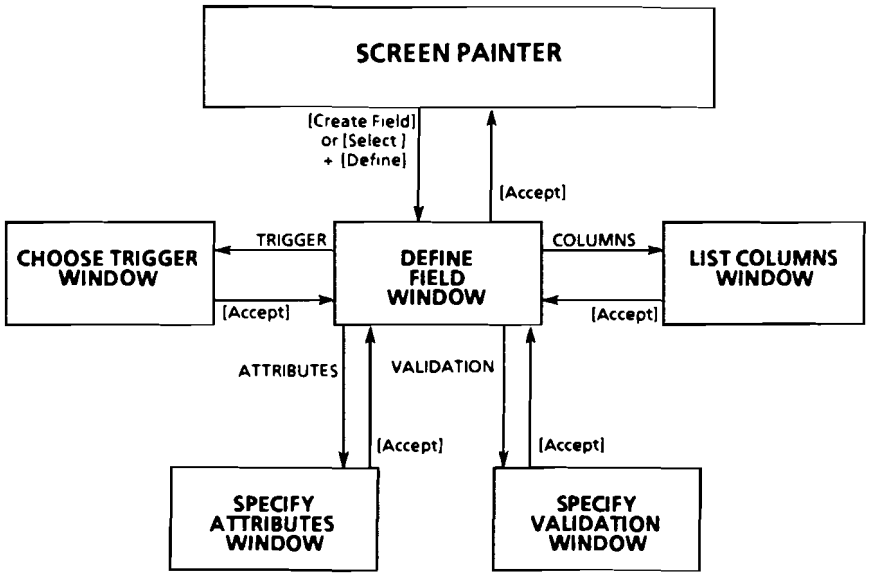
BIJLAGE 3. Overzicht van de menu's die gebruikt worden bij het onwerpen van een form met behulp van SQL*FORMS. (Overgenomen uit SQL*FORMS Designer's Reference)



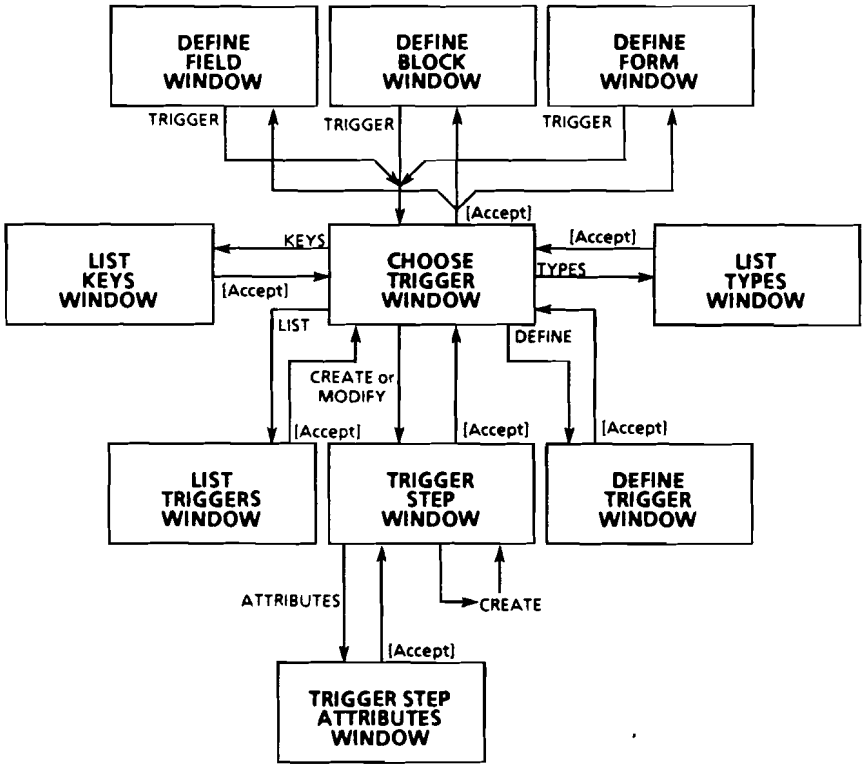
figuur B3.1. Menu's die gebruikt worden om een form te creëeren of te veranderen.



figuur B3.2. Menu's die gebruikt worden om een blok te definiëren.



figuur B3.3. Menu's die gebruikt worden om een veld te definiëren.



figuur B3.4. Menu's die gebruikt worden om een trigger te definiëren.