V. DATA DEFINITIE

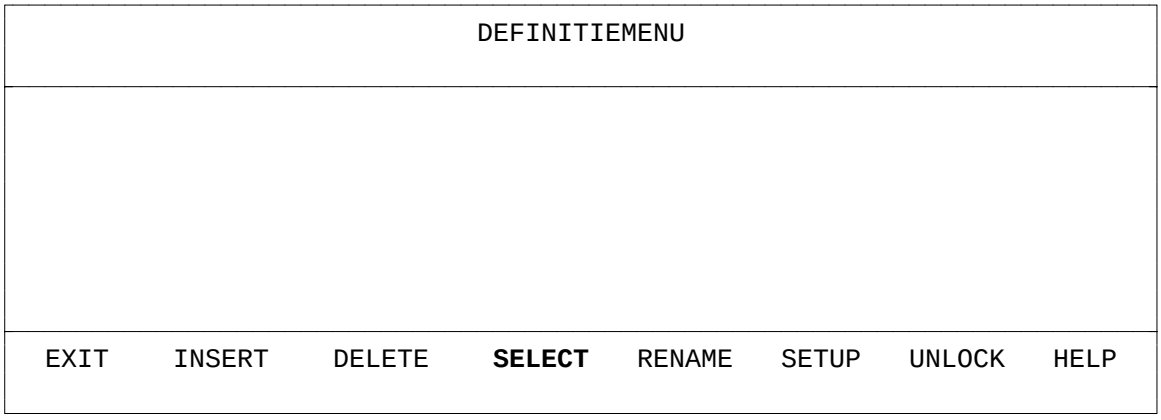
Wanneer voor het eerst gebruik gemaakt wordt van XPLAIN dan verschijnt in het HOOFDMENU een welkomtekst. Deze tekst geeft algemene informatie en wordt slechts eenmalig vertoond.

Vanuit het hoofdmenu zijn in eerste instantie alleen de opties DEFINE, REPORT en ACCESS beschikbaar. De opties **EXIT** (terug naar vorige menu) en **HELP** (voor raadpleging van de online handleiding) zijn op ieder menu beschikbaar en worden in het vervolg niet meer besproken.

De optie **DEFINE** is alleen beschikbaar indien men heeft ingelogt met **dba** of als lid van de DBA-groep (zie hoofdstuk VIII. AUTORISATIEMANAGER). Met **REPORT** kunnen rapporten worden aangemaakt (zie hoofdstuk IX. RAPPORTGENERATOR) en met **ACCESS** (zie hoofdstuk VIII. AUTORISATIEMANAGER) kunnen gebruikersrechten (ook o.a. passwords) worden ingesteld.

Als er databases aanwezig zijn verschijnen in het hoofdmenu ook de optie **BACKUP** waarmee een copie van een volledige database (incl. queries en panels) gemaakt wordt en de optie **RESTORE** waarmee een copie teruggezet wordt.

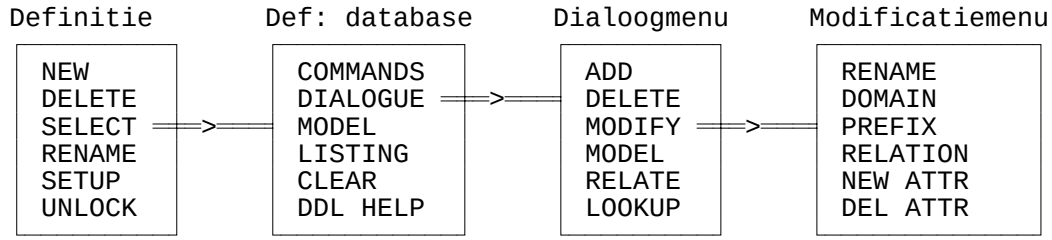
Na DEFINE gekozen te hebben komt men in het DEFINITIEMENU. Het is als volgt:



De menu-opties DELETE, SELECT, RENAME en UNLOCK zullen niet aanwezig zijn als er nog geen databases gedefinieerd zijn. Men kan dan INSERT kiezen om een nieuwe (lege) database toe te voegen. Alle andere menu-opties zullen zodra het zinvol is aanwezig zijn. Via SELECT kan men in het data definitie-menu komen waarbij men eerst een databasenaam moet selecteren uit de aanwezige databases die in het midden van het scherm getoond worden.

Hieronder wordt een overzicht gegeven van de menustructuur van de submenu’s waarbij

telkens de standaard menu-opties EXIT en HELP zijn weggelaten:



Vanuit het datadefinitie-menu (Definitie) kan men de database definiëren. Men heeft daarvoor de keuze uit twee verschillende mogelijkheden. De eerste is via opdrachten (COMMANDS) en de tweede is via menu's en vraagstellingen (DIALOGUE). **DIALOGUE** is vooral geschikt voor minder ervaren gebruikers omdat er steeds een dialoog plaats vindt tussen systeem en gebruiker waarbij alleen de relevante gegevens één voor één gevraagd worden aan de gebruiker. De gebruiker hoeft hiervoor niet de datadefinitietaal te leren. **COMMANDS** is vooral geschikt voor ervaren gebruikers. De gebruiker voert alle opdrachten in m.b.v. een tekstverwerker (editor) waarna alle opdrachten in één keer verwerkt worden. Hierbij is kennis van de datadefinitietaal (DDL) nodig. Overigens kunnen beide vormen ook door elkaar gebruikt worden.

Afgezien van de opties EXIT en HELP zullen de keuzemogelijkheden hierna toegelicht worden. Eerst worden van het database definitiemenu de opties INSERT, DELETE en RENAME behandeld. Tot slot behandelen we SELECT (V.4.) waarmee de daadwerkelijke datadefinitie van een database kan worden uitgevoerd.

V.1. **NEW** (voeg database toe)

Na de opeenvolgende keuzes DEFINE, NEW krijgt men het volgende scherm te zien:

```
                          DEFINITIEMENU

Gedefinieerde databases:
................

Naam van de toe te voegen database : ....................
```

Een databasenaam kan maximaal 20 karakters lang zijn. Alleen letters en cijfers zijn toegestaan. Met de <Return>-toets geeft men aan klaar te zijn met het intypen van de naam. Zolang niet op <Return> gedrukt is kan men nog wijzigingen in de naam aanbrengen (string

editor). Met de <PF1>-toets kan men de huidige naam leeg maken (wissen) of indien de naam leeg is de actie afbreken zodat er geen database toegevoegd wordt.

Databasenamen moeten uniek zijn. Indien men een reeds bestaande databasenaam toe wil voegen dan wordt dit niet geaccepteerd en verschijnt er een foutmelding.

Na het invoeren van een nieuwe databasenaam komt de gebruiker weer in het definitiemenu. Via SELECT kan de zojuist toegevoegde databasenaam geselecteerd worden en komt men in het datadefinitiemenu waar de databasestructuur gedefinieerd kan worden.

V.2. **DELETE** (verwijder database)

Na de keuzes DEFINE en DELETE worden alle aanwezige databases getoond in het midden van het scherm. De gebruiker moet m.b.v. de pijltjestoetsen de te verwijderen database aanwijzen. Hierbij wordt de actuele databasenaam in een speciale kleur (reversed video) aangegeven. Met de <Return>-toets kan men aangeven die database te verwijderen. Met de <PF1>-toets kan men de actie annuleren en keert men terug naar het Database Beheer menu. Als men met de <Return>-toets opdracht gegeven heeft de aangewezen database te verwijderen zal eerst gekontroleerd worden of dit toegestaan is. Zo ja, dan wordt er een extra bevestiging gevraagd aan de gebruiker. Dit ziet er als volgt uit:

```
                    DEFINITIE: VERWIJDEREN DATABASE

Selecteer database:

bibliotheek
tentamenverwerking


Bevestig de verwijdering van de database:                NEE  JA
```

Met een pijltjestoetsen kan NEE of JA geselecteerd worden, gevolgd door <Return>.

Mogelijke meldingen:

- **Database is niet verwijderd vanwege autorisatiereferenties.**
  In zo'n geval zijn er voor de betreffende database nog autorisaties voor typen in de database gedefinieerd. Deze moeten dan eerst verwijderd worden. Zie hiervoor hoofdstuk VIII. AUTORISATIEMANAGER.

- **Database is niet verwijderd vanwege .. panels en/of .. query(s).**

In zo'n geval zijn er voor de betreffende database applicatieschermen (panels) en/of queries gedefinieerd. Deze moeten dan eerst verwijderd worden. Zie hiervoor hoofdstuk VI. DATA MANIPULATIE en/of VII. APPLICATIEGENERATOR.

V.3. **RENAME** (wijzig databasenaam)

Na de opeenvolgende keuzes DEFINE en RENAME worden alle aanwezige databases getoond in het midden van het scherm. De gebruiker moet m.b.v. de pijltjestoetsen de te wijzigen databasenaam aanwijzen. Met de <Return>-toets kan men aangeven die databasenaam te willen wijzigen. Men komt dan in de string editor waarmee men de naam kan wijzigen. Na selectie van de databasenaam 'studenten' verschijnt onderaan het scherm:

```
Nieuwe database naam : bibliotheek.........
```

De originele naam wordt getoond zodat die eenvoudig is aan te passen. Met de <Return>-toets kan men het editen beëindigen. Met de <PF1>-toets (2x) kan men de wijziging annuleren.

Een (nieuwe) databasenaam moet uniek zijn. Hierop wordt gekontroleerd. Indien dit niet geldt voor de databasenaam dan zal er een foutmelding gegeven worden waarna men de mogelijkheid krijgt om de databasenaam als nog uniek te maken.

De foutmelding luidt:

- **Database is reeds gedefinieerd!**

V.4. **SELECT** (selecteer database)

Na de opeenvolgende keuzes DEFINE en SELECT worden alle aanwezige databases getoond in het midden van het scherm. Bijvoorbeeld:

```
                    DEFINITIE: SELECTEREN DATABASE

Selecteer database:

bibliotheek
tentamenverwerking


EXIT    NEW     DELETE    SELECT   RENAME    SETUP    UNLOCK    HELP
```

De gebruiker moet m.b.v. de pijltjestoetsen de gewenste database aanwijzen. Met de <Return>-toets kan men de selectie activeren waarna men in het Datadefinitie-menuscherm voor de desbetreffende database komt. Met de <PF1>-toets kan men de selectie annuleren en keert men terug naar het definitiemenu.

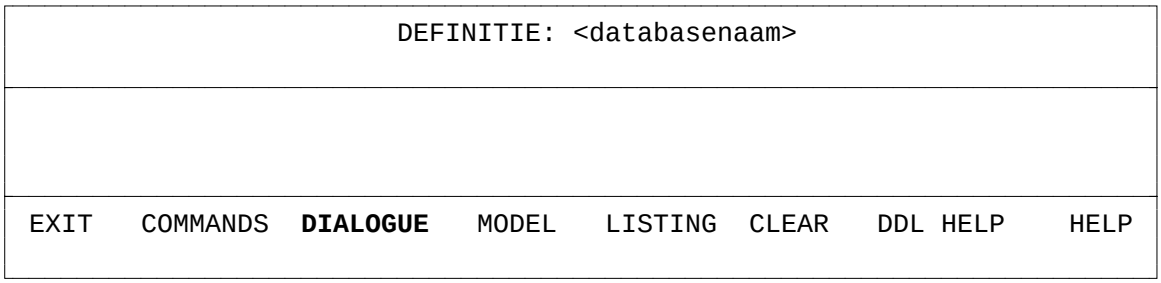
Database locking
Na het selecteren van een databasenaam kan de volgende melding verschijnen:

Database is reeds in gebruik door <login> op <systeem> sinds <datum en tijd>

Deze melding verschijnt als een (andere) gebruiker reeds gebruik maakt van de geselecteerde database. Het is met Xplain namelijk niet toegestaan dat meerdere gebruikers tegelijkertijd werken met dezelfde database, wel met verschillende databases. Daarvoor wordt iedere keer dat een gebruiker een database selecteert de database tijdelijk ontoegankelijk gemaakt (gelocked) voor andere gebruikers. Als de gebruiker de database weer verlaat wordt de lock weer opgeheven en is de database weer beschikbaar voor een andere gebruiker. Deze locking geldt voor de hele database inclusief alle gedefinieerde queries (opvragingen) en applicaties (panels). Een database wordt gelocked als de gebruiker een database wil definiëren (DEFINE, SELECT), hernoemen (DEFINE, RENAME), verwijderen (DEFINE, DELETE) of gebruiken (USE). Bij de melding wordt ook aangegeven wie op dat moment gebruik maakt van de database, sinds wanneer en op welke host (computer/werkstation).

UNLOCK (verwijder database lock)
Soms kan het voorkomen dat een database ten onrechte gelocked is doordat bijvoorbeeld de spanning voor de computer is weggevallen. De beheerder moet dan nagaan of de database inderdaad onterecht gelocked is (het programma XPLAIN draait niet meer). In dat geval kan iemand met dba-permissie de lock verwijderen door gebruik te maken van deze optie.

Na de selectie van een database komt men in het volgende Definitie-menu:



In dit scherm kan men kiezen uit twee vormen (COMMANDS en DIALOGUE), een overzicht van het databasemodel (MODEL), een verwerking van het databasemodel in een rapport (LISTING) het verwijderen van alle definities zodat er een lege database overblijft (CLEAR) of het raadplegen van de syntax van de definitie commando’s (DDL HELP). Daarnaast zijn natuurlijk ook de standaard opties EXIT en HELP aanwezig.

DIALOGUE is vooral geschikt voor minder ervaren gebruikers omdat er steeds een dialoog plaats vindt tussen systeem en gebruiker waarbij alleen de relevante gegevens één voor één gevraagd worden aan de gebruiker. De gebruiker hoeft hiervoor niet de datadefinitietaal te leren.

COMMANDS is vooral geschikt voor ervaren gebruikers. De gebruiker voert alle opdrachten in m.b.v. een tekstverwerker (editor) waarna alle opdrachten in één keer verwerkt worden. Hierbij is kennis van de datadefinitietaal (Data Definition Language = DDL) nodig. Men kan eventueel gebruik maken van de **DDL HELP** optie voor een beknopte samenvatting van de DDL syntax.

Overigens kunnen beide dialoogvormen ook door elkaar gebruikt worden. Een voorbeeld is DIALOGUE voor de typedefinities en als aanvulling hierop COMMANDS voor de initialisaties.

Eerst zal DIALOGUE (V.4.1.x.) behandeld worden. Daarna COMMANDS (V.4.2.x.), MODEL (V.4.3.), LISTING (V.4.4.) en CLEAR (V.4.5.).

V.4.1. DIALOGUE (interactieve datadefinitie)

Na selectie van DIALOGUE krijgt men het volgende menuscherm te zien:

```
                    TYPEDEFINITIE: <databasenaam>

EXIT     ADD     DELETE     MODIFY     MODEL     RELATE     LOOKUP     HELP
```

De vijf van belang zijnde opties - ADD, DELETE, MODIFY, MODEL, RELATE en LOOKUP - worden hierna behandeld.

V.4.1.1. ADD (toevoegen (basis)typen)

Na selectie van ADD (type) krijgt men het volgende scherm te zien:

```
                        TOEVOEGEN TYPEN

              Nieuwe typenaam: ....................
```

Aan de hand van voorbeelden zal het interactief definiëren toegelicht worden. Hoewel Xplain toestaat in willekeurige volgorde samengestelde en basis-typen te definiëren is het verstandig onder in de abstractie-hiërarchie te beginnen.
Hieronder volgt eerst de definitie van een aantal basistypen. De definitie van het basistype "aantal" verloopt als volgt (niet relevante beeldtekst is weggelaten):

```
                    Nieuwe typenaam: aantal.............
            Domein (a, b, i, r of d): i
              Maximum aantal cijfers: 4
{Hierna kan men onderin kiezen voor een extra domeinbeperking}
        Traject of opsomming? (t, o): t
Ondergrens (return bij min oneindig): 1200
    Bovengrens (return bij oneindig): 9000
          Naam van eerste attribuut:
{Op <PF1> of <Return> gedrukt i.v.m. basistype}

Gedefinieerd type:
base aantal(I4)(1200..9000)
```

Overigens is men niet verplicht een extra waardebeperking aan te geven. Hier is gekozen voor een integer-traject, in het volgende voorbeeld is bij het basistype kamersoort in hotel gekozen voor een integer-opsomming.
Via de keuze voor een **waardebeperking** kan men domeinen van toegestane waarden van attributen definiëren. De mogelijke waardebeperkingen zijn: "o" (opsomming), "t" (traject) of "p" (patroon). Voor integer-domein kan men kiezen uit (t,o). Voor alfanumeriek kan men uit (p,o) kiezen. Voor real-domein kan men alleen "t" kiezen. De beperking opsomming resulteert **altijd** in de definitie van een basistype.

```
                    Nieuwe typenaam: kamersoort.........
            Domein (a, b, i, r of d): i
              Maximum aantal cijfers: 3
{Hierna kan men onderin kiezen voor een extra domeinbeperking}
        Traject of opsomming? (t, o): o
                      Eerste element: 1..
                   Volgende element: 2..
                   Volgende element: 3..
                   Volgende element: 11.
                   Volgende element: 12.
                   Volgende element: ...
{Met <Return> opsomming gestopt}

Gedefinieerd type:
base kamersoort (I3) (Opsomming).
```

De volgende voorbeelden laten zien dat bij typen die alfanumeriek gerepresenteerd worden er ook waardebeperkingen kunnen worden vastgelegd (merk op dat opsomming basistype is):

```
                    Nieuwe typenaam: naam..............
            Domein (a, b, i, r of d): a
             Maximum aantal posities: 15
{Hierna kan men onderin kiezen voor een extra domeinbeperking}
        Patroon of opsomming? (p, o): o
                      Eerste element: horloge........
                   Volgende element: wekker.........
                   Volgende element: radio..........
                   Volgende element: ...............
{Opsomming gestopt met <Return>}

Gedefinieerd type:
base naam(A15)(opsomming).
```

Een voorbeeld van een alfanumeriek patroon bij "postcode":

```
                    Nieuwe typenaam: postcode............
            Domein (a, b, i, r of d): a
             Maximum aantal posities: 6.
{Hierna kan men onderin kiezen voor een extra domeinbeperking}
        Patroon of opsomming? (p, o): p
                             Patroon: 9999xx
               Naam eerste attribuut: ..............................
{Voor basistype op <PF1> of <Return> drukken}

Gedefinieerd type:
base postcode(A6)(9999xx)
```

Bij typen met als domein REAL (r) kan men uitsluitend een traject aangeven, met de de beperking dat onder- en bovengrens alleen **gehele** waarden kunnen zijn. Een real domein is per definitie basistype en dus niet samengesteld.

```
                    Nieuwe typenaam: prijs..............
            Domein (a, b, i, r of d): r
Maximum aantal cijfers voor de komma: 5.
            Maximum aantal decimalen: 2.
{Hierna kan men onderin kiezen voor een extra domeinbeperking}
Ondergrens (return bij min oneindig): 0....
    Bovengrens (return bij oneindig): 12500

Gedefinieerd type:
base prijs(R5,2)(0..12500).
```

Merk op dat bij definitie van een basistype met een boolean als domein niet naar eventuele attributen gevraagd wordt:

```
                    Nieuwe typenaam: betaald.............
            Domein (a, b, i, r of d): b

Gedefinieerd type:
base betaald(B).
```

Nadat tevens het basistype "gewicht" (R3,3) gedefinieerd is kan het samengestelde type "artikel" (A3) als volgt gedefinieerd worden:

```
                    Nieuwe typenaam: artikel.............
            Domein (a, b, i, r of d): a
             Maximum aantal posities: 3.
{Hierna kan men kiezen voor een extra waardebeperking}
               Naam eerste attribuut: naam............................
             Naam volgende attribuut: gewicht........................
             Naam volgende attribuut: inkoop_prijs...................
             Naam volgende attribuut: verkoop_prijs..................
             Naam volgende attribuut: ...............................
{Stoppen met <PF1> of <Return>}

Gedefinieerd type:
type artikel(A3) = naam(A15)(Opsomming), gewicht(R3,3),
     inkoop_prijs(R5,2)(0..12500), verkoop_prijs (R5,2) (0..12500).
```

Indien bij attributen een type genoemd wordt dat op dat moment nog niet gedefinieerd is dan zal het systeem eerst een specificatie vragen voor dat (basis)type. In de voorafgaande voorbeelden is dat niet het geval.

Alleen indien het domein van het geintroduceerde type voldoet aan de eisen van het domein van een identificatie (alfanumeriek en minder dan 15 posities, integer of date) zal het systeem om eventuele attributen vragen. Als hierbij met <Return> gereageerd worden, zal het systeem een basistype veronderstellen. In andere gevallen zal het systeem in een nesting van typedefinities raken. Het volgende is hiervan een eenvoudig voorbeeld.

Het volgende voorbeeld toont de dialoog indien een type wordt gedefinieerd dat attributen heeft van een nog niet gedefinieerd type. We gaan uit van de volgende definitie:

type voorraad = magazijn, artikel, aantal.

Te denken valt aan een winkelbedrijf met meerdere magazijnen. Hierbij zijn tot nu toe alleen "artikel" en "aantal" gedefinieerde typen, "magazijn" kan daarbij gedefinieerd worden als:

type magazijn = straat, nummer, telefoon.

```
                    Nieuwe typenaam: voorraad.....
            Domein (a, b, i, r of d): i
              Maximum aantal cijfers: 3
               Naam eerste attribuut: magazijn.....

 Type magazijn nog niet gedefinieerd.
            Domein (a, b, i, r of d): i
              Maximum aantal cijfers: 2
               Naam eerste attribuut: straat.....

 Type straat nog niet gedefinieerd.
            Domein (a, b, i, r of d): a
             Maximum aantal posities: 14
               Naam eerste attribuut: .....
 {Bovenstaande regel verdwijnt omdat <PF1> of <Return> is ingedrukt}

 Type straat is attribuut van:
 type magazijn(I2) = straat(A14).

              Naam volgende attribuut: nummer.....

 Type nummer nog niet gedefinieerd.
            Domein (a, b, i, r of d): a
             Maximum aantal posities: 4
               Naam eerste attribuut: .....
 {Bovenstaande regel verdwijnt omdat <PF1> of <Return> is ingedrukt}

 Type nummer is attribuut van:
 type magazijn(I2) = straat(A14), nummer(A4).

              Naam volgende attribuut: telefoon.....

 Type telefoon nog niet gedefinieerd.
            Domein (a, b, i, r of d): a
             Maximum aantal posities: 14
               Naam eerste attribuut: .....
 {Bovenstaande regel verdwijnt omdat <PF1> of <Return> is ingedrukt}

 Type telefoon is attribuut van:
 type magazijn(I2) = straat(A14), nummer(A4), telefoon(A14).

{Onderin verschijnt de volgende vraag:                              }
{Samenhang tussen magazijn en voorraad?      AGGREGATIE GENERALISATIE }

 Type magazijn is attribuut van:
 type voorraad(I4) = magazijn(I2).

              Naam volgende attribuut: artikel.....
{Onderin verschijnt de volgende vraag:                              }
{Samenhang tussen voorraad en artikel?       AGGREGATIE GENERALISATIE }

              Naam volgende attribuut: aantal.....
              Naam volgende attribuut: .....
 {Bovenstaande regel verdwijnt omdat <PF1> of <Return> is ingedrukt}

 Gedefinieerd type:
 type voorraad(I4) = magazijn(I2), artikel(A3), aantal(I4)(1200..9000).

                     Nieuwe typenaam: .....
 {Dialoogscherm verdwijnt omdat <PF1> of <Return> is ingedrukt}
```

De volgende foutmeldingen op typen zijn mogelijk:

- **Lengte moet liggen tussen 0 en 10.**
  Een integer mag hoogstens uit 9 cijfers bestaan. Dit geldt overigens ook voor het aantal cijfers achter de komma bij real-waarden.

- **Lengte moet liggen tussen 0 en 16.**
  Een real mag hoogstens 15 cijfers voor de komma hebben.

- **Lengte moet liggen tussen 0 en 81.**
  Een alfanumerieke waarde (een karakterstring) mag ten hoogste uit 80 tekens bestaan.

- **Bovengrens mag niet kleiner zijn dan ondergrens.**
  Een leeg traject is niet toegestaan.

- **Definitie type A = ..., A, ... is niet toegestaan.**
  Overigens mag deze definitie wel als het **attribuut** A een rol (prefix) krijgt.

- **Definitie type A = B, ..., B, ... is niet toegestaan.**
  Als één van de attributen B een prefix krijgt dan is dit wel toegestaan. Dit geldt ook bij twee verschillende prefixen.

- **Definitie type A = x_A, ..., x_A niet toegestaan.**
- **Definitie type A = x_B, ..., x_B niet toegestaan.**
  Indien men 2 of meer attributen van het zelfde type wil opnemen dan moeten deze verschillende prefixen hebben. Eventueel mag één van de prefixen ontbreken.

- **Definitie type A = ..., [A], ... is niet toegestaan.**
  Een type mag geen specialisatie van zichzelf zijn.

- **Definitie type A = [B], ..., [x_B], ... is niet toegestaan.**
  Een dubbele specialisatie van eenzelfde type, eventueel onderscheidbaar van elkaar door een rol, is niet zinvol.

- **Recursieve typedefinitie niet toegestaan.**
  Een recursieve definitie kan de volgende vorm hebben:
  type A = B, ...
  type B = C, ...
  type C = A, ...

- **Omkeerbaarheid geldt niet.**
  Er bestaat al een samengesteld type met dezelfde attributen.

- **Type kan niet meer worden toegevoegd.**
  Meer dan 300 typen is thans niet mogelijk.

- **Attribuut kan niet meer worden toegevoegd.**
  Het totaal aantal attributen mag thans niet groter dan 300 zijn.

Het systeem geeft tijdens dialogen niet altijd foutmeldingen. Voorbeelden zijn:

- Een (basis)typenaam mag geen under-score/line ('_') of spatie bevatten: het systeem reageert met een pieptoon.

- Domeinen afwijkend van (a, i, b of r) worden niet geaccepteerd, zodat een correcte keuze mogelijk blijft. Het systeem reageert ook hier met een pieptoon.

V.4.1.2. DELETE (verwijderen (basis)typen)

Na de opeenvolgende keuzes DIALOGUE, DELETE krijgt men het volgende scherm te zien:

```
                        VERWIJDEREN TYPE

        Het te verwijderen type: ...................
```

Men kan alleen typen verwijderen als de relateerbaarheid op typeniveau blijft gelden. Stel dat men de volgende definities in een model heeft:

type student = naam, adres, woonplaats, geboortedatum.
type inschrijving = datum, universiteit, student, studie.

Dan kan het type "student" niet verwijderd worden omdat er naar verwezen wordt vanuit het type "inschrijving". Het type "inschrijving" mag in dit model wel verwijderd worden. Omdat ook al de instances die bij het type horen, verloren gaan wordt eerst de definitie van het type gegeven en wordt daarna nog om een bevestiging van de verwijdering gevraagd, d.w.z. men kan nog kiezen tussen NEE en JA.

Indien er sprake is van referenties naar het ingetoetste (basis)type vindt verwijdering niet plaats en krijgt men de melding:

- **Er zijn attributen die refereren naar dit type.**

Soms zijn er autorisaties gedefinieerd op een (basis)type, dan wordt verwijderen geweigerd met als melding:

- **Er zijn autorisatiereferenties.**

Soms komt een (basis)type voor in een panel (applicatiescherm). Opdracht tot verwijdering resulteert in een weigering onder vermelding van de melding:

- **Er zijn referenties vanuit een panel.**

Nadat men bij de vraag om bevestiging voor JA heeft gekozen ziet men de bevestiging:

- **Type is verwijderd.**

Tracht men een niet gedefinieerd type te verwijderen dan luidt de foutmelding:

- **Ongedefinieerd type.**

V.4.1.3. MODIFY (wijzigen (basis)typen)

Het kan zijn dat bij nader inzien sommige definities herzien moeten worden. Na de opeenvolgende keuzes DIALOGUE, MODIFY krijgt men het volgende scherm te zien:

```
                    TYPEMODIFICATIE: <databasenaam>

EXIT   RENAME   DOMAIN   PREFIX  RELATION  NEW ATTR   DEL ATTR   HELP
```

De nu mogelijke menukeuzen worden hierna toegelicht met uitzondering van de standaard menu-opties EXIT en HELP.

V.4.1.3.1. RENAME (wijzigen (basis)typenaam)

Via deze keuze kunnen namen van typen veranderd worden. De te wijzigen naam moet binnen de betreffende database wel bekend zijn en de nieuwe naam mag nog niet voorkomen. Voldoet men hieraan niet dan ziet men passende foutmeldingen op het scherm.

Voorbeeldscherm:

```
                        WIJZIGEN TYPENAAM

              Bestaande typenaam: magazijn.............
                 Nieuwe typenaam: loods...............

Typenaam gewijzigd.                                          RETURN
```

Zolang men niet op <PF1> of <Return> drukt kan men doorgaan met het wijzigen van typenamen. Overigens blijft het domein bij naamswijziging onveranderd.

V.4.1.3.2. DOMAIN (wijzig domein en/of waardebeperking)

Via deze keuze kunnen domeinen van (basis)typen worden gewijzigd. Bij een basistype is dit alleen toegestaan indien het in geen enkel type gebruikt wordt als attribuut. Bij een samengesteld type is dit alleen toegestaan als het aantal instances van het betreffende type gelijk aan nul is.

Zolang men een wijziging afsluit met <Return> kan men doorgaan met wijzigen. Met <PF1> of <Return> direkt na de melding, keert men terug naar de voorafgaande menu.

Overigens kan na de keuze DOMAIN bij een niet te wijzigen domein een eerder gedefinieerde waardebeperking worden gewijzigd of weggelaten.

Bij trajecten kan men onder- en/of bovengrens veranderen; bij opsommingen kan men de opsomming uitbreiden met nieuwe waarden of een geheel nieuwe opsomming van waarden definiëren.

Voorbeeld van een dialoog op het scherm:

```
                        WIJZIGEN DOMEIN

            Bestaande typenaam: loods...............
       Domein (a, b, i, of d): a
     Maximum aantal posities: 3.
Patroon of opsomming? (p, o): p
                    Patroon: x99

Domein gewijzigd.                                            RETURN
```

Het kan zijn dat bij nader inzien de domein van "loods" toch aangepast moet worden. De dialoog is dan:

```
                          WIJZIGEN DOMEIN
_____________________________________________________________________________

            Bestaande typenaam: loods..............
       Domein (a, b, i, r of d): a
        Maximum aantal posities: 4.
   Patroon of opsomming? (p, o): o
               Eerste element: L001
             Volgende element: L002
             Volgende element: L003
             Volgende element: ....
{Bovenstaande regel verdwijnt omdat <PF1> of <Return> is ingedrukt}

Domein gewijzigd.                                                  RETURN
```

V.4.1.3.3. PREFIX (wijzigen rolnaam/prefix)

Na de keuze PREFIX kan een prefix van een attribuut gewijzigd of toegevoegd worden. Hieraan is echter één voorwaarde verbonden: het attribuut met de nieuwe naam mag niet reeds voorkomen als attribuutnaam bij het betreffende type.

Een voorbeeld van een dialoog:

```
                         WIJZIGEN PREFIX
_____________________________________________________________________________

                Bestaande typenaam: artikel.............
                     Attribuutnaam: inkoop_prijs .....................
Nieuwe prefix ('-' is geen prefix!): aankoop .............

Prefix gewijzigd.                                                  RETURN
```

Zodra men de typenaam heeft ingetypt verschijnt onderin de typedefinitie van het type als geheugensteuntje bij het wijzigen. Als men een wijziging eindigt met <Return> kan men nog een volgende wijziging definiëren. Zonder wijziging (dus alleen <PF1> of <Return>) kan bij een ander type een wijziging worden aangebracht. Opnieuw geldt nu dat met <PF1> of <Return> zonder nieuw type men terug gaat naar het voorafgaande typemodificatiemenu.

Er verschijnen foutmeldingen na intoetsen van een ongedefinieerde typenaam of attribuutnaam. Mogelijke foutmeldingen zijn:

- **Ongedefinieerd type.**
- **Basistype heeft geen attributen.**
  men heeft achter de typenaam de naam van een basistype ingetoetst.
- **Attribuut niet bij type gedefinieerd.**

V.4.1.3.4. RELATION (wijzigen specialisatie <-> aggregatie)

Via RELATION kan men een attribuut dat gedefinieerd is als specialisatie veranderen in een aggregatie. Het omgekeerde is ook mogelijk.

Na intypen van de betreffende typenaam en attribuutnaam vindt de omzetting van specialisatie naar generalisatie (of omgekeerd) plaats. Bij foutieve of ongedefinieerde namen verschijnt een passende foutmelding. Een voorbeeldscherm:

```
                          WIJZIGEN ABSTRACTIE

          Bestaande typenaam: artikel ............
        Te wijzigen attribuut: magazijn ..............................

Attribuut is gerelateerd aan specialisatie.                  RETURN
```

Zou "magazijn" van voorafgaand voorbeeld oorspronkelijk een specialisatie zijn dan zou men de bevestiging hebben gekregen: "Attribuut is gerelateerd aan aggregatie".

V.4.1.3.5. ADD (toevoegen van attribuut aan een type)

Via MODIFY, ADD kan men attributen toevoegen aan de definitie van samengestelde typen. Een voorwaarde is dat de naam van een toe te voegen attribuut nog niet voorkomt bij de bestaande attribuutnamen van het betreffende samengestelde type. Met een prefix (voor de rolnaam) kan men een attribuutnaam eventueel uniek maken. Een voorbeeldscherm:

```
                          TOEVOEGEN ATTRIBUUT

              Bestaande typenaam: artikel.............
                   Attribuutnaam: verkoop_prijs.......................

type artikel(I3)(1..500) = kleur(A9)(Opsomming),
     aankoop_prijs (R4,2)(0..9900), verkoop_prijs (R4,2)(0..9900).

Attribuut toegevoegd.                                        RETURN
```

Zodra men de typenaam heeft ingetypt verschijnt onderin de typedefinitie van het type als geheugensteuntje bij het wijzigen. Het kan zijn dat men tracht een attribuut toe te voegen van een nog niet gedefinieerd type. Men moet dan terug naar een eerder menu. Vanuit DEFINE kan men na INSERT gekozen hebben (basis)typen toevoegen.

Stoppen met attributen toevoegen kan d.m.v. de <PF1>-toets.

V.4.1.3.6. DELETE (verwijderen van attribuut uit een type)

Via MODIFY, DELETE kunnen attributen van typen verwijderd worden. Verwijderen van een attribuut wordt toegestaan als er minstens één attribuut na verwijdering over blijft. Het laatste/enige attribuut van een type kan alleen verwijderd worden indien er geen verwijzingen bestaan naar dit type. Dan wordt tevens het type verwijderd omdat het geen bestaansrecht meer heeft.

Een voorbeeld van een dialoog bij attribuutverwijdering:

```
                        VERWIJDEREN ATTRIBUUT


              Bestaande typenaam: artikel.............
                   Attribuutnaam: aankoop_prijs.......................

type artikel(I3)(1..500) = kleur(A9)(opsomming),
     verkoop_prijs(R4,2)(0..9900).

Attribuut verwijderd.                                              RETURN
N.B. Corresponderend type blijft aanwezig!
```

Zodra men de typenaam heeft ingetypt verschijnt onderin de typedefinitie van het type als geheugensteuntje bij het wijzigen. Na op <Return> gedrukt te hebben krijgt men de gelegenheid nog meer attributen van het zelfde samengestelde type te verwijderen. Indien gewenst wordt attributen van een ander type te verwijderen dan moet op <PF1> of <Return> gedrukt worden. Wil men stoppen met verwijdering van attributen dan moet nogmaals op <PF1> of <Return> gedrukt worden.

Er verschijnen passende foutmeldingen indien niet bestaande namen ingegeven zijn. Mogelijke foutmeldingen:

- **Ongedefinieerd type.**
  De naam van het samengestelde type is niet gedefinieerd.

- **Attribuut niet bij type gedefinieerd.**
  Het attribuut met de ingetoetste naam is niet gedefinieerd.

- **Er zijn referenties.**
  Het systeem weigert het enige attribuut van een samengesteld type te verwijderen omdat er referenties bestaan naar dit type.

V.4.1.3.7. MODEL (toon databasemodel)

Met de keuze MODEL krijgt men een overzicht van alle definities behorende bij de geselecteerde database. Getoond worden onder andere de basistypen inclusief hun waarde-beperking, typen (bij specialisatie staat de naam van de generalisatie tussen "[" en "]") en de waarden van opsommingen behorende bij basistypen. Daarnaast worden ook de nog niet behandelde inits en constanten getoond. Deze inits en constanten kunnen alleen via COMMANDS gedefinieerd worden.

Voorbeeld van een scherm met datadefinities:

```
                              MODEL: demo

base naam(A20).
base straat(A15).
base huisnummer(A3.
base plaats(A15).
base postcode(A6)(9999xx).
base prijs(R5,2).
base betaald(B).
type medewerker(I4) = naam(A20), straat(A15), huisnummer(A3),
     plaats(A15), postcode(A6)(9999xx).

  ** MEER **  Naar memo: ...................


  < zes regels voor memo >
```

Zolang er nog meer definities te tonen zijn wordt met <PF1> of <Return> verder gebladerd. Uiteindelijk zal men met <PF1> of <Return> het lijsten van de definities verlaten.

Achter "Naar memo:" kan men eventueel de naam van een type intoetsen. Deze typedefinitie zal tijdens COMMANDS, DELETE of MODIFY bruikbaar zijn als geheugensteun. De ruimte voor het kladblaadje is echter beperkt tot 6 regels. Een volgende ’gang’ door het model resulteert in een volledig nieuw memo, zodat de ruimte ten volle benut wordt.

V.4.1.4. MODEL (toon databasemodel)

Dit geeft hetzelfde resultaat als hierboven in V.4.1.3.7. is beschreven.

V.4.1.5. **RELATE** (toon gerelateerde typen)

Veronderstel dat men een type wil verwijderen. Deze actie kan pas worden uitgevoerd nadat alle referenties naar dit type zijn verwijderd. Deze keuze biedt de mogelijkheid om het gebruik van een bepaald type als attribuut in het conceptuele model te onderzoeken.

Voorbeeld:

```
                        GERELATEERDE TYPEN

             Bestaande typenaam: prijs.............

Gerelateerde typen van prijs:
type artikel = aankoop_prijs, ...
type artikel = verkoop_prijs, ...

              Naar memo: ...................

 <zes regels voor memo>
```

V.4.1.6. **LOOKUP** (toon onderliggende typen)

Deze keuze biedt de mogelijkheid om het gebruik van een bepaald type in het conceptuele model te onderzoeken. Er wordt een opsomming gegeven van de attributen (basis en samengestelde typen).

Voorbeeld:

```
                        ONDERLIGGENDE TYPEN

             Bestaande typenaam: verkoop............

Onderliggende typen van verkoop:
type verkart = ...
base hoeveelheid.
base bedrag.
type klant = ...
base datum.

              Naar memo: ...................

 <zes regels voor memo>
```

V.4.2. COMMANDS (datadefinitie d.m.v. opdrachten)

Na de opeenvolgende keuzes DEFINE, SELECT kan men in het Definitiemenu naast de dialoogvorm (DIALOGUE) ook de commandovorm (COMMANDS) kiezen. De hierbij gebruikte taal wordt DDL (Data Definition Language) genoemd. De syntax is achterin deze handleiding opgenomen, zie Appendices X.2. SYNTAX VAN DE XPLAIN DATATAAL.

Via de keuze COMMANDS komt u in de DDL-editor terecht:

```
                          DDL EDITOR: demo
 top:   1 line:   1                                              insert

<PF1> : EXIT               <PF2>   : LOAD COMMANDS   <PF3> : DELETE LINE
<PF4> : IMPORT COMMANDS    <ENTER> : SWITCH TYPEOVER/INSERT

 <zes regels voor memo>
```

In het middenscherm kunt u de datadefinitie-opdrachten invoeren en wijzigen. Indien reeds eerder met de DDL-editor is gewerkt dan zullen de opdrachten van die laatste keer weer te zien en te editen zijn. Iedere keer dat men de DDL-editor verlaat m.b.v. <PF1> zal het systeem de laatste toestand automatisch opslaan in het tekstbestand **ddl.inv**.

De editing-faciliteiten en -toetsen zijn reeds in hoofdstuk IV.4.3. beschreven. De DDL-editor heeft nog twee extra functietoetsen <PF3> en <PF4> beschikbaar. Met <PF3> kan men de regel waarop de cursor staat verwijderen. Met <PF4> kan men het scherm vullen met gegevens uit een van de gedefinieerde rapporten.

Met <PF2> kan men de ingetoetste opdrachten laten verwerken/uitvoeren (dit hoeven niet alle definitie-opdrachten te zijn, bijvoorbeeld de opdrachten die met dialogen zijn ingevoerd worden niet opnieuw geladen). Mogelijke foutmeldingen die gegeven worden tijdens verwerking worden behandeld in paragraaf V.4.2.9.

Nu volgt een behandeling van de opdrachten van de XPLAIN datadefinitietaal. Van elke opdracht wordt eerst aangegeven, waartoe de opdracht gebruikt kan worden. Vervolgens wordt de syntax van de opdracht gedefinieerd en tenslotte worden de bijzonderheden van de syntax besproken. Iedere opdracht moet worden afgesloten met een punt.

Voor de syntactische definitie wordt de volgende notatie gehanteerd:

- woorden die geheel in vet staan zijn de sleutelwoorden, en dienen door de gebruiker letterlijk te worden ingevoerd (voorbeelden: **base, type, init**);

- woorden die beginnen met een hoofdletter en vanaf de tweede letter uit kleine letters bestaan, zijn onderdeel van een opdracht (voorbeeld: Attribuutlijst);
  De gebruiker dient deze woorden te substitueren. De substitutie die uitgevoerd moet worden, wordt gedefinieerd met ':='. (voorbeeld: attribuutlijst := ...).

- woorden die geheel uit kleine letters bestaan, zijn willekeurige (door de gebruiker te kiezen) namen; bijvoorbeeld: naam van een type;

- speciale tekens moeten letterlijk overgenomen worden. Voorbeelden zijn: (,) en = ;

- een verticale streep geeft een keuzemogelijkheid aan; bijvoorbeeld:
  purge typenaam | **purge** basistypenaam ;

- een vierkante haak geeft een optie aan. De gebruiker kan dit gedeelte van de opdracht eventueel weglaten; bijvoorbeeld: [**default**];

V.4.2.1. Base-opdracht

Met de opdracht **base** kan een basistype gedefinieerd worden. Van het basistype dient de gebruiker naast de naam ook de domein op te geven. Optioneel is een waardebeperking of domeinrestrictie in de vorm van een traject, een patroon of een opsomming.

De syntax van de opdracht is:

base basistypenaam (domein) . |
base basistypenaam (domein) (ondergrens..bovengrens) . |
base basistypenaam (domein) "patroon" . |
base basistypenaam (domein) = Opsomming .

Domeinen bestaan uit een hoofdletter voor het datatype en eventueel een getal voor de maximale lengte van de data. De domein voor een karakterstring is (**Ax**), voor een integer (**Ix**), voor een boolean (**B**), voor een real (**Rx,y**) en voor een date (**D**). Hierin zijn x en y positieve getallen die de maximale datalengte aangeven.
Bijvoorbeeld:

A15 (string voor 15 karakters),
I4 (integer van 4 cijfers),
R3,2 (real met 3 cijfers voor en 2 cijfers achter de decimale punt),
B (boolean).
D (date)

Definitie van een basistype ’naam’, een string van maximaal 25 karakters, gaat als volgt:

base naam (A25).

Vergeet de punt aan het einde van de opdracht niet.

Indien er bij een integer of real een waardebeperking in de vorm van een traject moet worden opgegeven, kan dit door het opgeven van een onder- en bovengrens. Voorbeelden:

base korting (I2) (0..25).
base bedrag (R4,2) (0..5000).

Bij reals kunnen alleen integers voor de onder- en bovengrens gebruikt worden.

Ook kan er bij karakterstrings een waardebeperking in de vorm van een patroon worden opgegeven. Speciale tekens in patronen zijn:
- ? voor willekeurig karakter
- x voor willekeurige letter
- 9 voor willekeurig cijfer
- de tekens , . / \ ; : - en spatie die letterlijk moeten voorkomen in het patroon.

Voorbeeld:

base postcode (A7) "9999 xx".

Tenslotte is er nog de mogelijkheid om de mogelijke waarden die het type aan kan nemen, één voor één op te sommen. Deze mogelijkheid geldt alleen voor basistypen met domein I of A waarbij de lengte maximaal 15 is. De elementen van de opsomming moeten worden gescheiden door komma’s. Voor A-waarden geldt, dat zij door dubbele quotes omsloten dienen te zijn. Voorbeelden:

base werkdag (A2) = "ma", "di", "wo", "do", "vr".
base oneven (I1) = 1, 3, 5, 7, 9.

V.4.2.2. Type-opdracht

Met de opdracht **type** kunnen samengestelde typen gedefinieerd worden. Naast de naam en domein van het samengestelde type moeten ook de attributen worden gespecificeerd. Deze attributen kunnen basistypen of samengestelde typen zijn. Met behulp van een prefix kan een rolattribuut gespecificeerd worden. Tenslotte kan nog met behulp van vierkante haken rond een attribuut aangegeven worden, of het attribuut een specialisatie is of niet. De syntax van deze opdracht is:

type typenaam (domein) = Attribuutlijst .

Instances van een type worden geïdentificeerd d.m.v. hun identificatiewaarde. Men moet het domein van deze identificaties opgeven. Deze mag alleen I of A (tot maximaal 15 karakters) zijn. Een extra waardebeperking in de vorm van een opsomming is niet toegestaan.

Ieder samengesteld type moet tevens op zijn minst één attribuut bevatten. Voorbeelden:

type artikel (A6) = beschrijving, voorraad, prijs.
type werknemer (I4) = naam, adres, woon_plaats, salaris.
type ingenieur (I3) = [werknemer], studierichting, specialisatie, afstudeer_jaar.

V.4.2.3. Init-opdracht

De **init** restrictie is een expliciete dynamische restrictie die bij een **insert-operatie** een beperking oplegt aan één of meerdere attribuutwaarden van het samengestelde type. Deze beperking kan al dan niet afhankelijk zijn van de databasetoestand. Als de initialisatie slechts wordt uitgevoerd, als de gebruiker geen attribuutwaarde invoert, is er sprake van een **default** (verstek) restrictie. De default restrictie, die kan worden gezien als een speciale vorm van de init restrictie, wordt verder parallel aan de init restrictie behandeld.

Indeling van init restricties

De initwaarde van een attribuut wordt bepaald met behulp van een expressie. Logischerwijs worden de initialisaties dan ook ingedeeld naar de soorten expressies die de initwaarden kunnen bepalen.

De eerste indeling is in conditionele en onconditionele expressies. De conditionele expressie bevat een **if-then-else** expressie, een **case** expressie, of een combinatie van beide. Een onconditionele expressie bevat geen van deze expressies.

De tweede indeling is in eigenschap-expressies en constante-expressies. Bij een eigenschap-expressie is de initwaarde afhankelijk van één of meer attribuutwaarden. De initwaarde is dan afhankelijk van de databasetoestand. Dit in tegenstelling tot de constante-expressies, waarbij de initwaarde slechts bepaald wordt door constanten, en dus onafhankelijk van de database-toestand.

Combinatie van deze twee indelingen levert drie klassen van init restricties op, te weten:

1 inits afhankelijk van een onconditionele eigenschap-expressie;
2 inits afhankelijk van een conditionele eigenschap-expressie;
3 inits afhankelijk van een constante-expressie.

Init definitie

Om initialisaties te kunnen definiëren is de opdracht **init** beschikbaar. Het attribuut waarop

de initialisatie betrekking heeft dient reeds gedefinieerd te zijn met behulp van de opdracht **type** via COMMANDS of met behulp van DIALOGUE. De syntax van de **init** opdracht is:

init [**default**] typenaam **its** attribuutnaam **=** init-expressie .

De **default** versie van de **init** opdracht zorgt ervoor dat de initialisatie van het attribuut alleen gebeurt indien de gebruiker geen waarde voor het attribuut meegeeft tijdens de **insert**.

In de init-expressie mogen constanten, getallen, strings en attributen uit de ontledingsboom (d.w.z. specificaties van attributen m.b.v. de **its** constructie) voorkomen. Daarnaast is de **if-then-else** expressie en de **case** expressie toegestaan. De init-expressie kan dus één van de volgende expressies zijn:

1 De attribuut expressie waarin attributen uit de ontledingsboom van het betreffende type mogen voorkomen.

2 De **if-then-else** expressie met de volgende syntax:

if boolean-expressie **then** expressie **else** expressie

Het **else** gedeelte is verplicht, aangezien de initialisatie voor elke databasetoestand een waarde moet opleveren. De boolean-expressie kan alleen de waarde TRUE of FALSE aannemen.

3 De **case** expressie met de volgende syntax:

case attribuut **of**
labellijst **:** expressie ;
... ...
default **:** expressie

De labellijst is een lijst met constanten (strings of integers). Indien het attribuut een waarde heeft die in de labellijst voorkomt, wordt de bijbehorende expressie uitgevoerd. Het verplichte **default**-gedeelte zorgt er voor, dat bij elke databasetoestand een initwaarde berekend kan worden. De default-expressie wordt uitgevoerd als de attribuutwaarde niet in de labellijst voorkomt.

Stel dat de volgende typedefinities ten behoeve van een toepassing voor orderverwerking beschikbaar zijn:

type artikel = beschrijving, voorraad, prijs.
type bestelling = klant, datum, totaal_bedrag.
type bestelregel = bestelling, soort, aantal, artikel, beschrijving, korting, bruto_prijs, netto_prijs, bedrag.

Hierbij kunnen de volgende initialisaties gedefinieerd worden:

```
init default bestelregel its aantal = 1.
init bestelregel its beschrijving = artikel its beschrijving.
init bestelregel its bruto_prijs = artikel its prijs.
init bestelregel its korting =
      if aantal < 10 then 0 else
      if aantal < 100 then 10 else 20.
init bestelregel its netto_prijs =
      (100 - korting) * artikel its prijs / 100.
init bestelregel its bedrag = aantal * netto_prijs.
```

Om het gebruik van de case expressie te illustreren, volgt nog een mogelijke formulering van een restrictie voor de bepaling van een kortingspercentage (let wel: attribuut 'korting' mag in werkelijkheid **hoogstens één** restrictie hebben):

```
init bestelregel its korting =
      case soort of
      "extra" : 10;
      "super" : 20;
      default : 0.
```

Het case attribuut dient een direkt attribuut van het betreffende type te zijn; het is niet toegestaan een attribuut uit de ontledingsboom van het betreffende type te gebruiken (bijvoorbeeld bestelregel its bestelling its datum).

Controle op geldigheid

Initialisaties kunnen alleen betrekking hebben op attributen. Bovendien kan elk attribuut maximaal één initialisatie met zich meedragen.

Initialisaties die betrekking hebben op een attribuut dat zelf een samengesteld type is, zijn mogelijk. Men dient er echter rekening mee te houden, dat deze attributen al een inherente restrictie met zich meedragen (namelijk dat de attribuutwaarde gedefinieerd moet zijn).

Tenslotte is er nog het probleem van de circulaire definitie. Een init kan afhankelijk zijn van andere attributen. Deze attributen kunnen op hun beurt weer initialisaties met zich meedragen die direct of indirect afhankelijk zijn van het eerste attribuut.

Voorbeeld:

```
type X = A,B,C.
init X its A = B.
init X its B = A.
```

Om A te berekenen is de waarde van B nodig en andersom. In deze situatie wordt een foutmelding gegeven, aangezien het programma anders bij de berekening van één van deze initwaarden in een oneindige lus terecht zou komen.

V.4.2.4. Constant-opdracht

Om een constante voor opslag van een enkele waarde te definiëren is de opdracht **constant** beschikbaar. Van de constante dient zowel de naam als de domein opgegeven te worden. De syntax is:

constant constantnaam (domein) .

Voor 'domein' kunnen dezelfde specificaties gebruikt worden als bij de **base** opdracht. Waardebeperkingen zijn niet toegestaan. Voorbeelden:

constant BTWhoog (R2,1).
constant pi (R1,9).
constant tweepi (R1,9).
constant titel (A15).

Bij definitie worden de constanten geïnitialiseerd: 0 bij domein Integer of Real, TRUE bij domein Boolean en "" bij domein A.

V.4.2.5. Assign-opdracht

Nadat een constante gedefinieerd is moet een waarde hieraan kunnen worden toegekend. Dit kan met een **assign** opdracht. De syntax hiervan is:

constantnaam **= (**expressie**)** .

Deze expressie moet tijdens het verwerken van de DDL-opdrachten bepaald kunnen worden en mag dus geen attributen bevatten, of verwijzingen naar nog niet gedefinieerde constanten. **De waarde wordt tijdens data definitie eenmalig bepaald**. Alleen het resultaat wordt bewaard in de database en niet de expressie. Voorbeelden:

BTWhoog = (17.5).
pi = (3.14159265).
tweepi = (2 * pi).
titel = ("De heer/Mevrouw").

V.4.2.6. Expressies

Expressies worden gebruikt bij initialisaties en constanten. Het is daarom nuttig om de mogelijkheden van expressies in de DDL nader te bekijken.

De rekenkundige expressies kunnen als gebruikelijk worden ingevoerd. Er wordt rekening gehouden met de prioriteit en de complexiteit van de expressies wordt alleen beperkt door de grootte van het geheugen.

Naast rekenkundige expressie zijn toegestaan:
- stringfuncties (**combine**, **head** en **tail**) op attributen/constanten met domein alfa-numeriek (A);
- datumfuncties (**newdate**, **timedif**, **isdate**, **yearf**, **monthf**, **dayf** en **wdayf**) op attributen/ constanten met domein date (D);
- systeemvariabelen (**loginname** bij A en **systemdate** bij D).

Er zijn echter constructies die alleen in bepaalde expressies zijn toegestaan. Voor init-expressies zijn dit de **if-then-else** en de **case** expressies. De **if-then-else** expressies en **case** expressies kunnen in init-expressies genest worden, zowel onderling als met zichzelf.

In een aantal gevallen is slechts door de DDL-opdracht waarin zij gedefinieerd zijn, te onderscheiden wat toegestaan is. Het gebruik van attributen in constant-expressies wordt als een semantische fout gezien en wordt tijdens het parsen niet gecontroleerd.

Waar de grens tussen syntactische en semantische fouten ligt is in sommige gevallen moeilijk te definiëren. Het is mogelijk om in de syntax van de DDL op te nemen dat een string-constante en een geheel getal niet bij elkaar kunnen worden opgeteld. Door deze beperking niet in de syntax op te nemen is het syntactisch juist om 1 en "tien" bij elkaar op te tellen. De grens tussen syntax en semantiek ligt in dit geval dus bij de syntactische definitie.

V.4.2.7. Purge-opdracht

Deze opdracht draagt zorg voor de verwijdering van basistypen, samengestelde typen, attributen, initialisaties en constanten. De syntax is:

purge basistypenaam . |
purge typenaam . |
purge typenaam **its** attribuutnaam . |
purge init typenaam **its** attribuutnaam . |
purge constantenaam .

Het toevoegen van **init** aan een purge-opdracht is niet noodzakelijk, omdat een purge-opdracht zelf kan bepalen wat er verwijderd dient te worden. Als bijvoorbeeld bij een attribuut een initialisatie gedefinieerd is, dan moet altijd eerst de initialisatie verwijderd worden, voordat het attribuut zelf verwijderd kan worden.

Opmerking: Indien er verwijzingen bestaan naar een (basis)type dan wordt het (basis)type **niet** verwijderd. Ook bij verwijzingen vanuit een applicatiescherm zal een (basis)type of attribuut niet verwijderd kunnen worden.

V.4.2.8. #-opdracht (voor opnemen commentaar)

Door het gebruik van het #-teken kan men aangeven dat de tekst die achter het #-teken staat op die zelfde regel beschouwd moet worden als commentaar. Dit is erg handig als tijdens verwerking blijkt dat er ergens halverwege een fout geconstateerd is. De opdrachten tot aan die incorrecte opdracht zijn dan al wel verwerkt in de database. Een herhaalde aanbieding leidt tot foutmeldingen als "(Basis)type ... is al gedefinieerd". Door nu alle reeds succesvol uitgevoerde opdrachten vooraf te laten gaan door een # kan men ze laten staan zonder dat ze de volgende keer weer uitgevoerd gaan worden.

V.4.2.9. Foutmeldingen bij verwerking opdrachten

In paragraaf X.2. is de volledige syntax van de datadefinitietaal (DDL) opgenomen. Indien een opdracht niet aan deze syntax voldoet zal het programma een syntaxfout genereren. Mogelijke foutmeldingen zijn:

- **(Basis)type <naam> is al gedefinieerd.**
  Deze melding na ’base naam(A9).’, terwijl ’naam’ al gedefinieerd was.

- **(Basis)type kan niet worden toegevoegd.**
  Er kunnen bijvoorbeeld maximaal 300 (basis)typen gedefinieerd worden.

- **Onbekende Domein bij base <naam>.**
  De domein moet A, I, R of B zijn omringd door (en).

- **Lengte Integer/Real/String domein is te groot.**
  Integers kunnen maximaal 9 cijfers groot zijn, Reals maximaal 15 cijfers voor de decimale punt en 9 cijfers erachter waarbij het aantal voor en achter tezamen maximaal 15 mag zijn, Strings kunnen maximaal 80 karakters groot zijn.

- **Onbekend karakter in patroon bij base <naam>.**
- **Opsomming van reals is niet toegestaan. Base <naam>.**
- **Element in opsomming voldoet niet aan domein. Base <naam>.**
- **Element komt tweemaal in opsomming voor. Base <naam>.**

- **Onbekend karakter in domein.**
  De domein begint niet met een (.

- **Samengesteld type ’<typenaam>’ heeft een ongeldige domein.**
  De domein begint niet met een (, daarna een A of een I gevolgd een getal tussen de 1 en 15 en afgesloten door een).

- **Attribuut ’<naam>’ nog niet gedefinieerd als (basis)type.**
  Een attribuut moet eerst als basistype of type gedefinieerd worden voordat het gebruikt

kan worden in de attribuutlijst van een typedefinitie.

- **Attribuut '<naam>' komt twee maal voor in attributenlijst.**
- **Rolattribuut '<naam>' komt twee maal voor in attributenlijst.**
  Ieder attribuut moet een unieke naam hebben binnen de attributen van een type. M.b.v. een rol (prefix) kan een attribuutnaam uniek gemaakt worden. Als er al rollen gebruikt worden dan moet één van beide rollen veranderd worden om de attributen uniek te maken.

- **Attribuut '<naam>' is gelijk aan vader type.**
  De definitie: type persoon (A6) = naam, persoon. is niet toegestaan. Voeg een rol toe aan het attribuut persoon, dus: type persoon (A6) = naam, ouder_persoon.

- **Recursieve typedefinitie indien attribuut '<naam>' wordt toegevoegd.**
  Recursieve typedefinities over meerdere typen zijn niet mogelijk i.v.m. de relateerbaarheid. Recursie binnen één type is echter wel mogelijk.

- **Omkeerbaarheid geldt niet.**
  Typedefinities moeten uniek zijn. Zie paragraaf II.5. OMKEERBAARHEID.

- **Attribuuttabel is vol.**
  Er kunnen bijvoorbeeld maximaal 300 attributen (verdeeld over alle samengestelde typen) gedefinieerd worden.

- **Attribuut kan slechts 1 initialisatie met zich meedragen.**
  Een init definiëren voor een attribuut waarvoor reeds een init gedefinieerd is, is niet toegestaan. Verwijder eerst de huidige init (via purge init) om daarna de nieuwe init te kunnen definiëren.

- **Expressie type is niet compatibel met init attribuut type.**
  Een expressie resulteerd in een waarde van een bepaald datatype, bijvoorbeeld integer of boolean. Indien het attribuut van een ander datatype is dan de expressie dan is de init niet mogelijk. Wijzig de expressie of eventueel de domein van het attribuut.

- **Initialisatie resulteert in een circulaire definitie.**
  Circulaire definities zijn niet toegestaan. Verbreek de cirkel door één of meerdere init-expressies aan te passen.

- **Constante is al gedefinieerd.**
  Een constante met dezelfde naam bestaat reeds.

- **Onjuiste domein van constante.**
  Het domein van een variabele kan A, I, R, D of B zijn, gevolgd door de datalengte (behalve bij D en B). Een waardebeperking is niet mogelijk.

- **Constante kan niet worden toegevoegd.**
  Er kunnen maximaal 200 constanten gedefinieerd worden.

- **Constante '<constantenaam>' niet gevonden.**
  Er wordt verwezen naar een niet-gedefinieerde constante.

- **Arithmetische/Logische/Alfanumerieke expressie verwacht.**
  Bij een toekenning (assignment) van de waarde van een expressie aan een constante moet het datatype van de resultante van de expressie overeenkomen met de representaie van de constante.

- **Integer/Reëel getal past niet in domein.**
- **String (Alfanumerieke waarde) past niet in domein.**
  Waarde is groter (qua lengte) dan de maximale ruimte die tijdens definitie als domein is opgegeven.

- **Boolean kan slechts de waarde TRUE (1) of FALSE (0) aannemen.**
  Een toekenning van een Boolean-constante gaat als volgt: Betaald = (FALSE).

- **Purge: Type/Attribuut/Init/Constante niet gevonden.**
  De Init, Typenaam, Attribuutnaam of Constantenaam bestaat niet.

- **Type/Attribuut/Init/Constante kan niet verwijderd worden.**
  Vanwege verwijzingen naar het Type/Attribuut/Init/Constante is het niet mogelijk om het Type/Attribuut/Init/Constante te verwijderen. Haal eerst de verwijzing weg.

- **Attribuut niet verwijderd. Laatste attribuut van type.**
  Een verwijdering van het laatste attribuut zou betekenen dat de omkeerbaarheidseis niet meer geldt voor dat type. Men kan het type verwijderen door: purge <typenaam>.

- **Attribuut niet verwijderd. Samengesteld type heeft panel referenties.**
- **Attribuut niet verwijderd. Er zijn referenties vanuit een panel.**
  Indien een attribuut gebruikt wordt in een panel (eventueel als koppelattribuut) dan moet deze referentie(s) eerst verwijderd worden voordat het attribuut verwijderd kan worden.

- **Attribuut niet verwijderd. Er zijn referenties vanuit initialisaties.**
  Indien een attribuutnaam gebruikt wordt in een init-expressie dan moet eerst deze expressie aangepast of verwijderd worden voordat het attribuut verwijderd kan worden.

- **Constantetabel (vt) / Initialisatietabel (it) / Expressieboom (eb) / Expressie_stringtabel (eb_st) / Eigenschapslijst (pl) / Gebruikslijst (ul) / is vol. Waarschuw sysop.**
  De interne grenzen voor opslag van bepaalde metatabellen zijn bereikt. Deze grenzen zijn echter alleen door systeemprogrammeurs aanpasbaar, neem hiervoor kontakt op met

Uw XPLAIN leverancier.

- **Syntax fout: Ongeldig keyword: 'insert'.**
  Datamanipulatie-opdrachten kunnen niet gebruikt worden tijdens datadefinitie.

V.4.3. MODEL (toon databasemodel)

Met de keuze MODEL krijgt men een overzicht van alle definities behorende bij de geselecteerde database. Getoond worden de definities van alle basistypen (inclusief hun waardebeperking), de typen (bij specialisatie staat de naam van de generalisatie tussen "[" en "]"), de inits, de waarden van opsommingen en de constanten met hun waarde. Zie voor een voorbeeldscherm paragraaf V.4.1.3.7.

V.4.4. LISTING (verwerking van databasemodel in rapport)

Via deze optie kan men het model opnemen in een rapport. De juiste vorm en volgorde waarin definities in de listing gegeven worden maken eventuele import-model eenvoudiger.

V.4.5. CLEAR (verwijderen van alle datadefinities)

Via de keuze CLEAR is men in staat het conceptuele model (**en alle bijbehorende database-gegevens**) te verwijderen. De datadefinitie-opdrachten blijven daarentegen wel in de DDL-editor bestaan zodat snel de hele database opnieuw gedefinieerd kan worden.

Nadat men CLEAR heeft gekozen wordt om een extra bevestiging gevraagd. Indien men de keuze JA maakt krijgt men het volgende scherm te zien:

```
                    Definitie van <databasenaam>

Gehele database wordt verwijderd!  Doorgaan?              NEE   JA
Alleen eventuele DDL_commando's blijven aanwezig.

Constanten worden verwijderd ....
Initialisaties worden verwijderd ....
Samengestelde typen worden verwijderd ....
Basistypen worden verwijderd ....
Definitie tabellen worden verwijderd ....
Autorisatie tabellen worden verwijderd ....

 EXIT  COMMANDS  DIALOGUE   MODEL   LISTING  CLEAR   DDL HELP    HELP
```

Na afloop komt men weer terug in het definitiemenu.