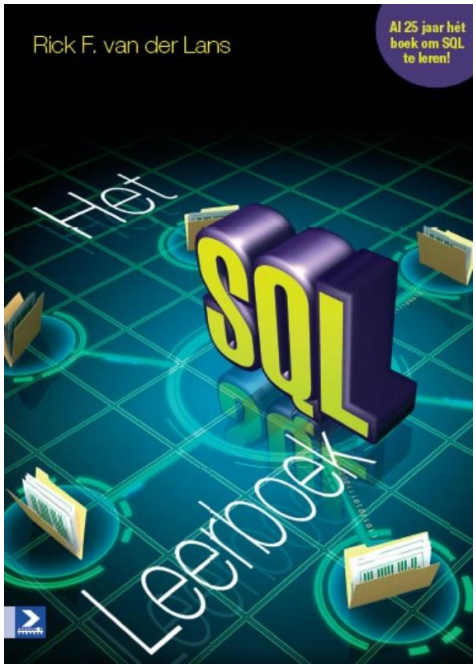
Het SQL Leerboek – zevende editie
De Relationele Algebra en Calculus



Auteur: Rick F. van der Lans
Versie: 1.0
Datum: Februari 2012

Hoofdstuk 1

Inleiding

1.1 Inleiding

Eén van de vormen van manipulatie is het selecteren van gegevens en hiervoor kent het relationele model *expressies*. Een expressie is te vergelijken met een zoekopdracht in een databasetaal, bijvoorbeeld: 'Zoek alle werknemers die in Voorburg wonen'. Het resultaat van een expressie is een relatie. Er zijn twee verschillende, maar gelijkwaardige 'stijlen' voor het formuleren van expressies:

- *Relationele algebra*; dit is een verzameling operatoren (gebaseerd op bewerkingen uit de verzamelingenleer), waarbij elke operator één of meer relaties als invoer heeft en een relatie als uitvoer. Door verschillende operatoren in een expressie te combineren, kan in principe elke vraag geformuleerd worden.
- *Relationele calculus*; met een formule (gebaseerd op de predikatenlogica) wordt gespecificeerd uit welke rijen het resultaat moet bestaan.

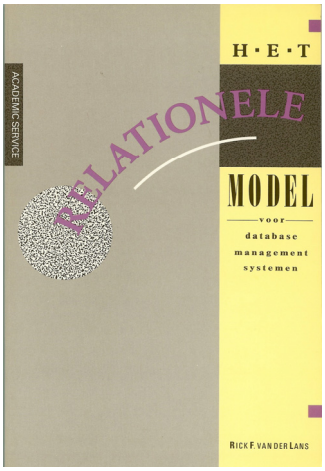
Bewezen is dat de kracht van de relationele algebra en die van de relationele calculus gelijk zijn. Er bestaat een algoritme dat algebraïsche expressies omzet naar calculus-expressies en er bestaat een algoritme waarmee calculus-expressies omgezet kunnen worden naar algebraïsche expressies. Dit document bevat een beschrijving van de relationele algebra en van de relationele calculus.

Indien u vragen of opmerkingen heeft op dit document hebt, laat het ons weten via email adres sql@r20.nl

Opmerking: Deze tekst in dit document is een bewerking van een hoofdstuk uit mijn boek *Het Relationele Model voor Database Management Systemen* dat in 1988 is uitgegeven, maar dat niet meer beschikbaar is; zie figuur 1.1. Omdat het voor sommigen toch interessant is om te zien wat de achtergrond van SQL is, is besloten deze tekst hier opnieuw beschikbaar te maken.

1.2 Beschrijving van het werknemersvoorbeeld

In dit document maken we veelvuldig gebruik van een voorbeeld dat betrekking heeft op werknemers, afdelingen en de cursussen die binnen een bedrijf gegeven worden. We noemen dit het *werknemersvoorbeeld*. In deze paragraaf geven we aan uit welke attributen en rijen de relaties bestaan, welke domeinen er zijn en geven we een toelichting op het geheel. Wij raden u aan het voorbeeld goed te bestuderen voordat u aan de volgende hoofdstukken begint.



Figuur 1.1 *De kaft van het boek 'Het Relationele Model voor Database Management Systemen'*

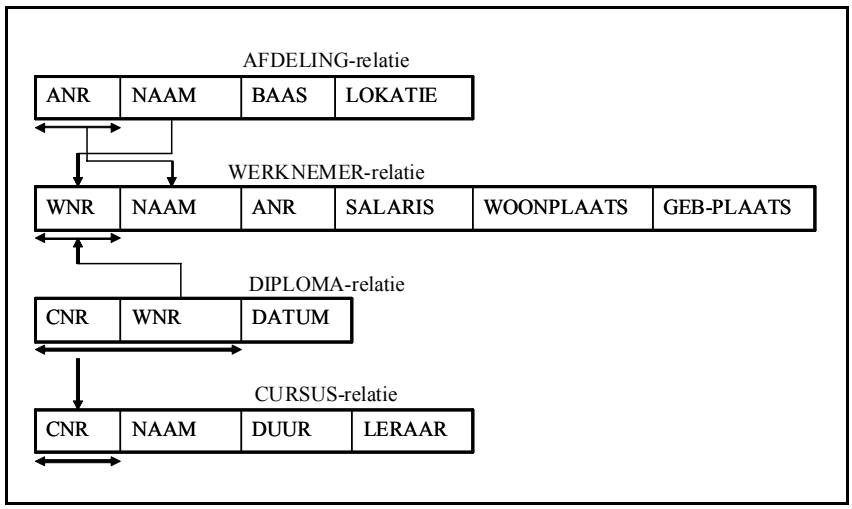
Van elke werknemer in het bedrijf wordt een aantal gegevens in de WERKNEMER-relatie geregistreerd, zoals een uniek nummer, de achternaam, het nummer van de afdeling waar de werknemer werkt, het jaarsalaris, de woonplaats en de geboorte plaats. Van sommige werknemers is de geboorteplaats onbekend.

Van elke afdeling in het bedrijf wordt in de AFDELING-relatie een uniek nummer, de naam van de afdeling, het nummer van de baas en de lokatie van de afdeling geregistreerd. Elke baas is op zich weer een werknemer en komt dus voor in de WERKNEMER-relatie. Als een werknemer baas is van een bepaalde afdeling wordt hij of zij ook beschouwd als werknemer van die afdeling.

In het bedrijf worden cursussen aan werknemers gegeven. In de CURSUS-relatie wordt geregistreerd welke cursussen gegeven kunnen worden. Van elke cursus wordt een uniek nummer, de naam, de tijdsduur en de naam van de leraar geregistreerd.

Tenslotte wordt in de DIPLOMA-relatie aangegeven op welke datum een werknemer voor een bepaalde cursus geslaagd is. De datum die geregistreerd wordt geeft aan wanneer het diploma uitgereikt is. De eerste twee cijfers geven het jaar aan, de middelste twee de maand en de laatste twee de dag van de maand.

Tot slot, in figuur 1.2 zijn de relaties met een strokendiagram grafisch weergegeven.



Figuur 1.2 *Strokendiagram van het werknemersvoorbeeld*

1.3 Databaseschema voor het werknemersvoorbeeld

Het databaseschema bevat geen relatietypes.

```
DATABASESCHEMA WERKNEMERS

SCHEMA (

DOMAIN WNR
   TYPE       NUMERIC
   POPULATION (0:25)
   ORDERING   JA

DOMAIN WNAAM
   TYPE       ALPHANUMERIC
   POPULATION (CHAR(10))
   ORDERING   JA

DOMAIN ANR
   TYPE       NUMERIC
   POPULATION (0:30000)
   ORDERING   JA

DOMAIN GELD
   TYPE       NUMERIC
   POPULATION (0:∞)
   ORDERING   JA

DOMAIN PLAATS
   TYPE       ALPHANUMERIC
   POPULATION ('Rotterdam', 'Amsterdam', 'Voorburg', 'Wassenaar', 'Wateringen', 'Rijswijk',
               'Amstelveen', 'Schiedam', 'Den Haag', 'Leiden')
   ORDERING   JA

DOMAIN ANAAM
   TYPE       ALPHANUMERIC
   POPULATION ('Verkoop', 'Inkoop', 'Magazijn', 'Planning')
   ORDERING   JA

DOMAIN CNR
   TYPE       NUMERIC
   POPULATION (0:30000)
   ORDERING   JA

DOMAIN CNAAM
   TYPE       ALPHANUMERIC
   POPULATION (CHAR(10))
   ORDERING   JA

DOMAIN TIJDSDUUR
   TYPE       NUMERIC
   POPULATION (0:100)
   ORDERING   JA

DOMAIN LERAAR
   TYPE       ALPHANUMERIC
   POPULATION (CHAR(10))
   ORDERING   JA
```

```
DOMAIN DATUM
   TYPE       NUMERIC
   POPULATION (750000:900000)
   ORDERING   JA

RELATION WERKNEMER
   SCHEMA ( WNR         WNR     NOT NULL ,
            NAAM        WNAAM   NOT NULL ,
            ANR         ANR     ,
            SALARIS     GELD    NOT NULL ,
            WOONPLAATS  PLAATS  NOT NULL ,
            GEB-PLAATS  PLAATS  )
   PRIM   ( WNR )
   REF    ( ANR --> AFDELING.ANR )

RELATION AFDELING
   SCHEMA ( ANR         ANR     NOT NULL ,
            NAAM        ANAAM   NOT NULL ,
            BAAS        WNR     NOT NULL ,
            LOKATIE     PLAATS  NOT NULL )
   PRIM   ( ANR )
   REF    ( WNR --> WERKNEMER.WNR )

RELATION CURSUS
   SCHEMA ( CNR         CNR     NOT NULL ,
            NAAM        CNAAM   NOT NULL ,
            DUUR        TIJDSDUUR   NOT NULL ,
            LERAAR      LERAAR  NOT NULL )
   PRIM   ( CNR )

RELATION DIPLOMA
   SCHEMA ( CNR         CNR     NOT NULL ,
            WNR         WNR     NOT NULL ,
            DATUM       DATUM   NOT NULL )
   PRIM   ( CNR, WNR )
   REF    ( CNR --> CURSUS.CNR )
   REF    ( WNR --> WERKNEMER.WNR )
```

)

1.4 Inhoud van de relaties van het werknemersvoorbeeld

Met opzet is gekozen voor relaties met een 'kleine' inhoud. Hierdoor worden de uitwerkingen van de vele voorbeelden in de volgende hoofdstukken niet te uitgebreid.

De WERKNEMER-relatie:

WNR	NAAM	ANR	SALARIS	WOONPLAATS	GEB-PLAATS
10	Dingelam	104	70000	Amsterdam	Amsterdam
15	Homme	102	60000	Wateringen	Rotterdam
25	Hondijk	108	40000	Voorburg	?
45	Alberegt	102	75000	Schiedam	Wassenaar
70	Osewoudt	108	42000	Voorburg	Rijswijk
80	Geyerstein	104	62000	Amstelveen	?
90	Hermans	108	55000	Rijswijk	Amsterdam

De AFDELING-relatie:

ANR	NAAM	BAAS	LOKATIE
102	Verkoop	45	Rotterdam
104	Inkoop	10	Amsterdam
108	Magazijn	90	Voorburg

De CURSUS-relatie:

CNR	NAAM	DUUR	LERAAR
10	Verkopen	2	Palm
20	Presenteren	4	Prater
30	Organiseren	4	Cools
40	Statistiek	10	Veen

De DIPLOMA-relatie:

CNR	WNR	DATUM
10	15	851110
10	45	851104
30	10	861220
30	15	850921
30	45	850422
40	90	860514
20	15	860630
40	15	850731

Hoofdstuk 2

De relationele calculus

2.1 Inleiding

De relationele algebra kent een groot aantal bewerkingen of *operatoren*. Sommige daarvan komen overeen met bewerkingen uit de verzamelingenleer en andere zijn speciaal 'uitgevonden' voor de relationele algebra. Vijf operatoren die in dit document besproken worden zijn *elementair*. De rest zijn *afgeleide* operatoren[1]. De elementaire operatoren zijn:

- **select-operator**; hiermee worden rijen in een relatie geselecteerd die aan een bepaalde conditie voldoen
- **project-operator**; hiermee worden bepaalde attributen in een relatie geselecteerd
- **product-operator**; hiermee wordt het cartesisch product van relaties gecreëerd
- **union-operator**; hiermee worden rijen van verschillende relaties samengevoegd
- **difference-operator**; hiermee worden rijen in een relatie geselecteerd die niet in een andere relatie voorkomen

Afgeleide operatoren zijn gedefinieerd in termen van de elementaire operatoren. De belangrijkste reden waarom afgeleide operatoren bestaan is dat het formuleren van sommige expressies lastig en omslachtig is, wanneer alleen van de elementaire operatoren gebruik gemaakt wordt. We behandelen de afgeleide operatoren separaat in hoofdstuk 3.

Er bestaan eigenlijk maar weinig relationele databasetalen die volledig op de relationele algebra gebaseerd zijn. Eén hiervan is ISBL (Information System Base Language). ISBL is de databasetaal van het database management systeem genaamd PRTV (Peterlee Relational Test Vehicle). PRTV[2] is ontwikkeld in het IBM UK Scientific Centre in Peterlee, County Durham in Engeland. Het is, net als het bekende System R, een systeem dat in een research-omgeving ontwikkeld is. Dit gehele project is reeds gestopt.

Ook al bestaan er dan maar weinig databasetalen gebaseerd op de algebra, toch wordt zij vaak gebruikt om de kracht van relationele databasetalen vast te stellen.

---

[1] Codd definieerde in zijn eerste artikel [CODD70] tevens drie andere operatoren: *permutatie*, *compositie* en *restrictie*. Deze operatoren zijn niet zo bekend geworden. We laten ze daarom in dit document buiten beschouwing.

[2] Schmidt J.W en Brodie M.L., *Relational Database Systems: Analysis and Comparison*, Springer-Verlag, 1983.

2.2 De algebraïsche expressie

Wat is een expressie? Een *expressie* is een samenstelling van relaties en operatoren met als *waarde* een relatie. Een expressie is te vergelijken met wat in het dagelijks leven een wiskundige formule wordt genoemd. Een formule heeft pas waarde als de waarden van alle onbekende variabelen ingevuld zijn.

```
<expressie> ::=
   <relatienaam> |
   ( <rij> { , <rij> } ) |
   <expressie> WHERE <conditie> |
   <attributenlijst> OF <expressie> |
   <expressie> <algebra-operator> <expressie> |
   <join-operator> ( <expressie>, <expressie> ) |
   <expressie> [ <attributen-lijst> ] |
   ( <expressie> )

<synoniem> ::= SYNONYM <synoniemnaam> : <relatienaam>

<rij> ::= < <domeinnaam>:<waarde> { , <domeinnaam>:<waarde> } >

<conditie> ::=
   <attribuut> <vergelijkings-operator> <waarde> |
   <attribuut> <vergelijkings-operator> <attribuut> |
   <attribuut> IS NULL |
   <conditie> <boolean-operator> <conditie> |
   NOT <conditie> |
   ( <conditie> )

<attributenlijst> ::= <attribuut> { , <attribuut> }

<attribuut> ::=
   <attribuutnaam> |
   <relatienaam>.<attribuutnaam> |
   <relatienaam>.<attribuutnaam>*

<waarde> ::=
   <numerieke-waarde> |
   <alfanumerieke-waarde>

<vergelijkings-operator> ::= = | < | > | >= | <= | <>

<algebra-operator> ::=
   TIMES |UNION | MINUS | INTERSECT | DIVISION | OUTER-UNION | OUTER-MINUS | OUTER-INTERSECT

<join-operator> ::=
   JOIN | MAYBE-JOIN | OUTER-JOIN | LEFT-OUTER-JOIN | RIGHT-OUTER-JOIN | GT-JOIN | LT-JOIN | GE-JOIN |
   LE-JOIN | NE-JOIN | NATURAL-JOIN

<boolean-operator> ::= AND | OR
```

De meest eenvoudige vorm van de expressie bestaat uit alleen de naam van een relatie. Voorbeeld:

```
( AFDELING )
```

De waarde van deze expressie is een relatie die uit dezelfde attributen en rijen bestaat als de AFDELING-relatie.

```
ANR  NAAM         BAAS  LOKATIE
---  -----------  ----  ---------
102  Verkoop        45  Rotterdam
104  Inkoop         10  Amsterdam
108  Magazijn       90  Voorburg
```

Een andere vorm van de expressie bestaat uit één of meer rijen, waarbij elke rij hetzelfde aantal attributen heeft. Voorbeeld:

```
( <NAAM:'Joyce', AFDELING:'Verkoop', PLAATS:'Rotterdam'>,
  <NAAM:'Diane', AFDELING:'Inkoop',  PLAATS:'Amsterdam'> )
```

Waarde:

```
NAAM   AFDELING  PLAATS
-----  --------  ---------
Joyce  Verkoop   Rotterdam
Diane  Inkoop    Amsterdam
```

De relationele algebra wordt een *gesloten model* (closure property) genoemd, omdat een algebraïsche expressie als 'invoer' 'en als 'uitvoer' relaties heeft. Omdat het resultaat van een expressie ook weer een relatie is, worden expressies weleens *expressierelaties* genoemd. Relaties zoals gedefinieerd in hoofdstuk 1, noemen we *basisrelaties*. Wat zijn nu de verschillen tussen expressie- en basisrelaties?

- Een basisrelatie heeft een inhoud, een expressierelatie niet. Beide hebben wel een waarde. De waarde van een basisrelatie is altijd gelijk aan haar inhoud. Een expressierelatie daarentegen heeft geen inhoud, de waarde wordt afgeleid van de waarden van één of meer basisrelaties.
- Een expressierelatie is afgeleid van één of meer basisrelaties. Een basisrelatie, de naam zegt het al, is niet afgeleid, maar is wat betreft inhoud en waarde onafhankelijk van andere relaties.
- Een expressierelatie hoeft niet te voldoen aan alle eisen die voor basisrelaties gelden. De enige eis die voor expressierelaties geldt, is dat een expressierelatie geen identieke rijen kan en mag bevatten.

In de volgende paragrafen worden één voor één de andere vormen van de expressie behandeld. Bij een aantal hiervan hebben we voor de duidelijkheid een deel van de definitie van de expressie gekopieerd.

2.3 Select-operator: selecteren van rijen

Met de *select-operator* worden rijen in een relatie geselecteerd die aan een bepaalde conditie voldoen. Het resultaat van de select-operator is een *horizontale deelverzameling* van een relatie. (De select-operator wordt ook wel de *restrict-operator* genoemd.)

```
<select-operator> ::= ( <expressie> WHERE <conditie> )
```

Met een *conditie* wordt aangegeven welke rijen uit de relatie van de genoemde expressie geselecteerd moeten worden. Een simpele conditie bestaat uit een attribuut gevolgd door een vergelijkingsoperator (zie tabel 2.1 voor de betekenis van elke operator) en gevolgd door een attribuut of een waarde.

Vergelijkingsoperator	Betekenis
=	gelijk aan
>	groter dan
<	kleiner dan
>=	groter dan of gelijk aan
<=	kleiner dan of gelijk aan
<>	ongelijk aan

Tabel 2.1 *De vergelijkingsoperatoren*

Voorbeelden van correcte condities zijn:

- `NAAM = 'Dingelam'`
- `WNR > 78`
- `WOONPLAATS <> GEB-PLAATS`

Voorbeeld 2.1: Geef de werknemers die in Voorburg wonen.

```
(WERKNEMER WHERE WOONPLAATS = 'Voorburg')
```

Resultaat:

```
WNR  NAAM         ANR  SALARIS  WOONPLAATS  GEB-PLAATS
---  -----------  ---  -------  ----------  ----------
 25  Hondijk      108    40000  Voorburg    ?
 70  Osewoudt     108    42000  Voorburg    Rijswijk
```

Omdat dit de eerste expressie is die in dit boek behandeld wordt, geven we een uitgebreide toelichting. Ten eerste, we maken onderscheid tussen expressies en operatoren. Een expressie is een samenstelling van één of meer operatoren. In het bovenstaande voorbeeld is de expressie uit slechts één operator samengesteld: de select-operator. In de volgende paragrafen zullen we expressies behandelen die uit meer operatoren bestaan. Ten tweede, we zullen het resultaat van een expressie in tabelvorm weergeven. De alfanumerieke waarden in een tabel worden links aangeschoven en de numerieke waarden worden rechts aangeschoven.

Maar wat betekent de expressie in feite? Met de expressie wordt gevraagd naar alle rijen uit de WERKNEMER-relatie waarin de waarde in het WOONPLAATS-attribuut gelijk is aan Voorburg.

Voorbeeld 2.2: Geef de werknemers die *niet* in Rotterdam geboren zijn.

```
(WERKNEMER WHERE GEB-PLAATS <> 'Rotterdam')
```

Resultaat:

```
WNR  NAAM         ANR  SALARIS  WOONPLAATS  GEB-PLAATS
---  -----------  ---  -------  ----------  ----------
 10  Dingelam     104    70000  Amsterdam   Amsterdam
 45  Alberegt     102    75000  Schiedam    Wassenaar
 70  Osewoudt     108    42000  Voorburg    Rijswijk
 90  Hermans      108    55000  Rijswijk    Amsterdam
```

Het resultaat bevat géén werknemers waarvan de geboorteplaats onbekend is. De vraag is of deze werknemers ook in het resultaat opgenomen zouden moeten worden. Waarschijnlijk wel als de NULL-waarde in dit attribuut betekent 'werknemer heeft geen woonplaats', maar in alle andere gevallen niet. En omdat we niet precies weten wat de NULL-waarde aangeeft, valt de NULL-waarde buiten het resultaat. In paragraaf 2.8 laten we zien hoe rijen, waar een attribuut de NULL-waarde bevat, toch geselecteerd kunnen worden.

Voorbeeld 2.3: Geef de werknemers die in hun geboorteplaats wonen.

```
(WERKNEMER WHERE WOONPLAATS = GEB-PLAATS)
```

Resultaat:

```
WNR  NAAM         ANR  SALARIS  WOONPLAATS  GEB-PLAATS
---  -----------  ---  -------  ----------  ----------
 10  Dingelam     104    70000  Amsterdam   Amsterdam
```

Let goed op bij het vergelijken van attributen op de mogelijke aanwezigheid van NULL-waarden. Dit resultaat bevat dus géén werknemers waarvan de geboorteplaats onbekend is. Ook de volgende expressie levert een resultaat op dat op het eerste gezicht niet verwacht wordt.

```
(WERKNEMER WHERE GEB-PLAATS = GEB-PLAATS)
```

Resultaat:

```
WNR  NAAM         ANR  SALARIS  WOONPLAATS  GEB-PLAATS
---  -----------  ---  -------  ----------  ----------
 10  Dingelam     104    70000  Amsterdam   Amsterdam
 15  Homme        102    60000  Wateringen  Rotterdam
 45  Alberegt     102    75000  Schiedam    Wassenaar
 70  Osewoudt     108    42000  Voorburg    Rijswijk
 90  Hermans      108    55000  Rijswijk    Amsterdam
```

Toelichting: In het resultaat ontbreken alle rijen waar het GEB-PLAATS-attribuut gelijk is aan de NULL-waarde. Dit komt omdat een NULL-waarde niet gelijk is aan een andere NULL-waarde. De waarde van de conditie is daarom onbekend.

Het is toegestaan om vóór de naam van een attribuut de naam van de relatie te specificeren. Attribuutnaam en relatienaam moeten door een punt gescheiden worden. De volgende expressie is dus gelijkwaardig aan de expressie in voorbeeld 2.2.

```
(WERKNEMER WHERE WERKNEMER.GEB-PLAATS <> 'Rotterdam')
```

Het expliciet noemen van de relatienaam is in deze expressie weliswaar overbodig. Er bestaat geen enkele twijfel over de relatie waartoe het GEB-PLAATS-attribuut behoort, want er is maar één relatie in de expressie gespecificeerd. In paragraaf 2.6 behandelen we expressies met meer relaties; daar is het toevoegen van relatienamen wel noodzakelijk om dubbelzinnigheden te voorkomen.

Indien in een conditie twee attributen met elkaar vergeleken worden, zoals in voorbeeld 2.3, gelden de volgende regels:

- Twee attributen mogen met = en <> vergeleken worden als ze op hetzelfde domein gedefinieerd zijn.
- Twee attributen mogen met <, >, >= en <= vergeleken worden als ze op hetzelfde domein gedefinieerd zijn èn als het domein met `ORDERING JA` gedefinieerd is.

De conditie in voorbeeld 2.3 is goed, omdat de twee attributen WOONPLAATS en GEB-PLAATS beide op het domein PLAATS gedefinieerd zijn. Maar de conditie `WNR = ANR` is ongeldig, omdat de attributen op verschillende domeinen gedefinieerd zijn. Pas als twee attributen op hetzelfde domein gedefinieerd zijn, zijn ze *vergelijkbaar*.

Laten we nog even terugkeren op het al dan niet geordend zijn van waarden in een domein. Als een domein gedefinieerd is als ongeordend, dan mogen de attributen die op dit domein gedefinieerd zijn alleen in condities gebruikt worden waar of het gelijk-aan teken of het ongelijk-aan teken gebruikt wordt. Kleiner-dan is bijvoorbeeld niet toegestaan, omdat wanneer van een bepaalde ordening sprake is, een bepaalde waarde alleen kleiner (of groter) dan een andere waarde is. Zo is het bijvoorbeeld niet mogelijk attributen die op het domein KLEUR zijn gedefinieerd met elkaar te vergelijken.

Net als de predikatenlogica kent de relationele algebra de operatoren ^, V en ¬. Wij geven in de relationele algebra deze operatoren met respectievelijk de woorden AND, OR en NOT weer. Een bewering in de predikatenlogica kan twee waarden aannemen: waar of onwaar. Er bestaat ook een variant van de predikatenlogica waarbij de waarde van een bewering drie waarden kan aannemen: waar, onwaar of *onbekend*. De relationele algebra is gebaseerd op deze speciale variant van de predikatenlogica. Als bijvoorbeeld twee condities met de AND-operator gecombineerd worden, ontstaat één van de drie waarden. De waarde hiervan is uiteraard afhankelijk van de waarden van de twee afzonderlijke condities. Wat die waarde is, is uit de waarheidstabel in tabel 2.2 af te leiden (A en B zijn calculusformules).

A	B	A AND B	A OR B	NOT A
waar	waar	waar	waar	onwaar
waar	onwaar	onwaar	waar	onwaar
waar	onbekend	onbekend	waar	onwaar
onwaar	waar	onwaar	waar	waar
onwaar	onwaar	onwaar	onwaar	waar
onwaar	onbekend	onwaar	onbekend	waar
onbekend	waar	onbekend	waar	onbekend
onbekend	onwaar	onwaar	onbekend	onbekend
onbekend	onbekend	onbekend	onbekend	onbekend

Tabel 2.2 *De waarheidstafel*

Voorbeeld 2.4: Geef alle werknemers die meer dan € 40.000,- verdienen en in Voorburg wonen.

```
(WERKNEMER WHERE SALARIS > 4000
           AND   WOONPLAATS = 'Voorburg')
```

Alle regels die betrekking hebben op de evaluatie van condities, zoals evaluatie van links naar rechts en de haakjes, zijn hier ook van toepassing.

2.4 Project-operator: selecteren van attributen

Met de *project-operator* worden bepaalde attributen in een relatie geselecteerd. Het resultaat van de project-operator wordt een *verticale deelverzameling* van een relatie genoemd.

```
<project-operator> ::= ( <attributenlijst> OF <expressie> )
```

Voorbeeld 2.5: Geef het nummer en de naam van elke werknemer.

```
(WNR, NAAM OF WERKNEMER)
```

Resultaat:

```
WNR  NAAM
---  ----------
 10  Dingelam
 15  Homme
 25  Hondijk
 45  Alberegt
 70  Osewoudt
 80  Geyerstein
 90  Hermans
```

Twee opmerkingen:

- De volgorde van de attributen in de project-operator mag afwijken van de volgorde waarin de attributen in de relatie gedefinieerd zijn.
- Identieke rijen worden automatisch uit het resultaat van een expressie verwijderd. De reden is duidelijk, het resultaat van een expressie is wederom een relatie en relaties bevatten nooit twee dezelfde rijen.

Voorbeeld 2.6: Geef de verschillende geboorteplaatsen.

```
(GEB-PLAATS OF WERKNEMER)
```

Resultaat:

```
GEB-PLAATS
----------
Amsterdam
Rotterdam
?
Wassenaar
Rijswijk
?
```

Toelichting: Het aantal rijen in dit resultaat is niet gelijk aan het aantal rijen in de WERKNEMER-relatie. Door het gebruik van de project-operator zijn alle identieke rijen uit het resultaat verwijderd. Plaatsnaam Amsterdam komt bijvoorbeeld tweemaal voor in het GEB-PLAATS-attribuut, maar is slechts éénmaal afgedrukt. Het resultaat bevat wel twee rijen met een NULL-waarde. Dit komt (nogmaals) omdat twee NULL-waarden niet aan elkaar gelijk zijn.

2.5 Combineren van operatoren met nesting

De voorbeelden van expressies die in de vorige twee paragrafen behandeld zijn, bestonden allemaal uit slechts één operator. De meeste vragen zijn echter niet met één operator te formuleren. Het is bijvoorbeeld onmogelijk om met één select-operator of één project-operator een expressie te formuleren die antwoord geeft op de volgende vraag:

Voorbeeld 2.7: Geef het nummer en de naam van elke werknemer die in Voorburg woont.

Wel is het uiteraard mogelijk met behulp van een assignment-expressie in twee stappen het gewenste resultaat te verkrijgen. Eerst worden alle werknemers die in Voorburg wonen geselecteerd. Dit resultaat wordt vervolgens toegekend aan de HELP-relatie:

```
HELP := (WERKNEMER WHERE WOONPLAATS = 'Voorburg')
```

Vervolgens halen we de attributen WNR en NAAM uit de HELP-relatie:

```
(WNR, NAAM OF HELP)
```

Resultaat:

```
WNR  NAAM
---  -----------
 25  Hondijk
 70  Osewoudt
```

Een andere en kortere manier is door meer operatoren in een expressie te combineren door middel van het *nesten van operatoren*. De formulering voor voorbeeld 2.7 ziet er dan als volgt uit:

```
(WNR, NAAM OF (WERKNEMER WHERE WOONPLAATS = 'Voorburg'))
```

Toelichting: Het verschil tussen de bovenstaande expressie en de expressies in de vorige paragrafen is dat op de plaats waar over het algemeen de naam van een (basis-)relatie staat nu een select-operator is ingevuld. Wat betekent dit precies? De project-operator wordt niet uitgevoerd op een bestaande relatie, maar op het resultaat van de select-operator. Het resultaat van de select-operator is uiteraard een relatie bestaande uit attributen en rijen. Dit resultaat wordt echter niet zichtbaar, maar vormt een zogenaamd *tussenresultaat*.

Het nesten van operatoren komt uit de wiskunde. De berekening 4 + (2 * 10) bevat ook een tussenresultaat: 20 (het resultaat van de vermenigvuldiging 2 * 10). Het getal 4 wordt bij dit tussenresultaat opgeteld. Het eindresultaat is 24 en *niet* 60.

Voorbeeld 2.8: Geef de naam, de woonplaats en het salaris van elke werknemer die in Voorburg woont en meer dan € 40.000,- verdient.

```
(NAAM, WOONPLAATS, SALARIS OF
   ((WERKNEMER WHERE SALARIS > 40.000)
               WHERE WOONPLAATS = 'Voorburg'))
```

Resultaat:

```
NAAM         WOONPLAATS  SALARIS
-----------  ----------  -------
Osewoudt     Voorburg      42000
```

Toelichting: Met de 'binnenste' select-operator worden de werknemers gezocht die meer dan € 40.000,- verdienen. Het resultaat hiervan is een horizontale deelverzameling van de WERKNEMER-relatie. Met de andere select-operator wordt in die deelverzameling gezocht naar alle werknemers die in Voorburg wonen. Het eindresultaat bevat de NAAM, het SALARIS en de WOONPLAATS van elke werknemer die meer dan € 40.000,- verdient en in Voorburg woont.

De instructie kan uiteraard ook geformuleerd worden door gebruik te maken van boolean-operatoren:

```
(NAAM, WOONPLAATS, SALARIS OF
   (WERKNEMER WHERE SALARIS > 40000
              AND   WOONPLAATS = 'Voorburg'))
```

Nog een laatste opmerking over het nesten van operatoren. Het relationele model stelt uiteraard geen limiet aan het nesten van operatoren. In principe kent het model geen enkele limiet. Er bestaat bijvoorbeeld geen limiet voor het maximum aantal attributen in een relatie, voor de maximale lengte van een alfanumerieke waarde, of voor de maximale grootte van een numerieke waarde. Vergeet niet dat het relationele model geen softwarepakket is, maar een theoretisch model!

2.6 Product-operator: cartesisch product van relaties

Met de product-operator wordt het cartesisch product van twee relaties gecreëerd.

```
<product-operator> ::= ( <expressie> TIMES <expressie> )
```

Als twee relaties met elkaar vermenigvuldigd worden, wordt elke rij uit de ene relatie 'geplakt' aan elke rij uit de andere relatie. Het aantal attributen in het resultaat (de graad) van de product-operator is gelijk aan de som van het aantal attributen in de ene relatie plus het aantal attributen in de andere relatie. Het aantal rijen in het resultaat (het kardinaalgetal) is gelijk aan het aantal rijen in de ene relatie vermenigvuldigd met het aantal rijen in de andere relatie. Het resultaat van de volgende expressie is hieronder weergegeven.

```
(DIPLOMA TIMES CURSUS)
```

Resultaat:

```
CNR  WNR  DATUM   CNR  NAAM         DUUR  LERAAR
---  ---  ------  ---  ----------   ----  ------
 10   15  851110   10  Verkopen        2  Palm
 10   15  851110   20  Presenteren     4  Veen
 10   15  851110   30  Organiseren     4  Cools
 10   15  851110   40  Statistiek     10  Veen
 10   45  851104   10  Verkopen        2  Palm
 10   45  851104   20  Presenteren     4  Veen
 10   45  851104   30  Organiseren     4  Cools
 10   45  851104   40  Statistiek     10  Veen
 30   10  861220   10  Verkopen        2  Palm
 30   10  861220   20  Presenteren     4  Veen
 30   10  861220   30  Organiseren     4  Cools
 30   10  861220   40  Statistiek     10  Veen
  :    :  :         :  :              :  :
  :    :  :         :  :              :  :
```

Elke rij in de DIPLOMA-relatie wordt zoveel maal afgedrukt als er rijen in de CURSUS-relatie zijn.

Indien meer dan twee relaties vermenigvuldigd moeten worden, moeten meer product-operatoren gebruikt worden. De volgende expressie heeft bijvoorbeeld het product van vier relaties als resultaat.

```
(WERKNEMER TIMES (AFDELING TIMES (CURSUS TIMES DIPLOMA)))
```

De haakjes kunnen in deze expressie weggelaten worden, omdat de product-operator *associatief* is:

```
(DIPLOMA TIMES CURSUS TIMES AFDELING TIMES WERKNEMER)
```

Het nut van de product-operator wordt pas duidelijk als zij gecombineerd wordt met andere operatoren. We zullen hier een aantal voorbeelden behandelen.

Voorbeeld 2.9: Geef van elke werknemer het nummer en de naam van de afdeling waarvoor hij of zij werkt.

```
(WNR, AFDELING.NAAM OF
   ((WERKNEMER TIMES AFDELING) WHERE
       WERKNEMER.ANR = AFDELING.ANR))
```

Resultaat:

```
WNR  NAAM
---  --------
 10  Verkoop
 15  Verkoop
 25  Magazijn
 45  Verkoop
 70  Magazijn
 80  Inkoop
 90  Magazijn
```

Toelichting: Eerst wordt de WERKNEMER-relatie met de AFDELING-relatie vermenigvuldigd. Het tussenresultaat is hieronder (gedeeltelijk) weergegeven.

```
WNR  NAAM         ANR  ...  ANR  NAAM      BAAS  ...
---  -----------  ---  ---  ---  -------   ----  ---
 10  Dingelam     104  ...  102  Verkoop     45  ...
 10  Dingelam     104  ...  104  Inkoop      10  ...
 10  Dingelam     104  ...  108  Magazijn    90  ...
 15  Homme        102  ...  102  Verkoop     45  ...
 15  Homme        102  ...  104  Inkoop      10  ...
 15  Homme        102  ...  108  Magazijn    90  ...
 25  Hondijk      108  ...  102  Verkoop     45  ...
 25  Hondijk      108  ...  104  Inkoop      10  ...
 25  Hondijk      108  ...  108  Magazijn    90  ...
 45  Alberegt     102  ...  102  Verkoop     45  ...
 45  Alberegt     102  ...  104  Inkoop      10  ...
 45  Alberegt     102  ...  108  Magazijn    90  ...
  :  :            :    :    :    :            :  :
  :  :            :    :    :    :            :  :
```

Vervolgens worden alle rijen in dit tussenresultaat geselecteerd waar de waarden in de twee ANR-attributen gelijk zijn. Omdat het tussenresultaat van de product-operator twee attributen met de naam ANR bevat, moet in de conditie aangegeven worden welk ANR-attribuut met welk ANR-attribuut vergeleken wordt, zodat hier geen misverstanden over bestaan. We geven dit aan door vóór de attribuutnaam de relatienaam te specificeren. De select-operator levert het volgende tussenresultaat op:

```
WNR  NAAM         ANR  ...  ANR  NAAM      BAAS  ...
---  -----------  ---  ---  ---  -------   ----  ---
 10  Dingelam     104  ...  104  Inkoop      10  ...
 15  Homme        102  ...  102  Verkoop     45  ...
 25  Hondijk      108  ...  108  Magazijn    90  ...
 45  Alberegt     102  ...  102  Verkoop     45  ...
 70  Osewoudt     108  ...  108  Magazijn    90  ...
 80  Geyerstein   104  ...  104  Inkoop      10  ...
 90  Hermans      108  ...  108  Magazijn    90  ...
```

Met de project-operator worden tenslotte de gevraagde attributen geselecteerd (zie de eerste tabel). Ook hier moet weer de relatienaam vóór het NAAM-attribuut gespecificeerd worden. Indien in plaats van AFDELING.NAAM WERKNEMER.NAAM gebruikt wordt, ontstaat het volgende resultaat:

```
WNR  NAAM
---  -----------
 10  Dingelam
 15  Homme
 25  Hondijk
 45  Alberegt
 70  Osewoudt
 80  Geyerstein
 90  Hermans
```

Een relatie mag ook met zichzelf vermenigvuldigd worden.

Voorbeeld 2.10: Geef de naam en het salaris van elke werknemer die meer verdient dan Hermans.

```
SYNONYM HERMANS : WERKNEMER

(WERKNEMER.NAAM, WERKNEMER.SALARIS OF
   ((WERKNEMER TIMES (HERMANS WHERE NAAM = 'Hermans'))
      WHERE WERKNEMER.SALARIS > HERMANS.SALARIS))
```

Resultaat:

```
NAAM         SALARIS
-----------  -------
Dingelam       70000
Homme          60000
Alberegt       75000
Geyerstein     62000
```

Toelichting: Eerst wordt voor de WERKNEMER-relatie een synoniem gedeclareerd, in dit geval het synoniem HERMANS. Met de product-operator wordt achter elke rij uit de WERKNEMER-relatie nog een rij geplakt die de gegevens bevat van Hermans. Dit tussenresultaat ziet er als volgt uit:

```
WNR  NAAM         SALARIS  ...  NAAM        SALARIS  ...
---  -----------  -------  ---  ----------  -------  ---
 10  Dingelam       70000  ...  Hermans       55000  ...
 15  Homme          60000  ...  Hermans       55000  ...
 25  Hondijk        40000  ...  Hermans       55000  ...
 45  Alberegt       75000  ...  Hermans       55000  ...
 70  Osewoudt       42000  ...  Hermans       55000  ...
 80  Geyerstein     62000  ...  Hermans       55000  ...
 90  Hermans        55000  ...  Hermans       55000  ...
```

Uit dit tussenresultaat worden alle rijen geselecteerd waar het linker SALARIS-attribuut groter is dan het rechter SALARIS-attribuut. Tussenresultaat:

```
WNR  NAAM         SALARIS  ...  NAAM        SALARIS  ...
---  -----------  -------  ---  ----------  -------  ---
 10  Dingelam       70000  ...  Hermans       55000  ...
 15  Homme          60000  ...  Hermans       55000  ...
 45  Alberegt       75000  ...  Hermans       55000  ...
 80  Geyerstein     62000  ...  Hermans       55000  ...
```

Het eindresultaat is nu eenvoudig te bepalen.

Zonder het gebruik van een synoniem is deze expressie niet te formuleren. Want veronderstel dat we het synoniem niet zouden gebruiken. Met de product-operator wordt dan de WERKNEMER-relatie met zichzelf vermenigvuldigd; dit levert nog geen problemen op. Maar hoe moeten nu in de select-operator de twee SALARIS-attributen uit elkaar gehouden worden? Dit kan niet zonder het gebruik van een synoniem. In feite is het begrip synoniem voor dit soort expressies in de relationele algebra opgenomen.

Een belangrijk verschil tussen de select- en project-operator enerzijds en de product-operator anderzijds is dat de eerste twee operatoren slechts één relatie als 'invoer' hebben en de product-operator twee. Hierdoor worden de select- en project-operatoren ook wel *unaire operatoren* genoemd en de product-operator een *binaire operator*.

2.7 Union-operator: samenvoegen van relaties

Met de *union-operator* (vereniging) worden rijen van een relatie met rijen van een andere relatie samengevoegd tot één relatie. Dubbele rijen worden uiteraard uit het resultaat verwijderd. Het aantal attributen in het resultaat is gelijk aan het aantal attributen van een van de twee relaties.

```
<union-operator> ::= ( <expressie> UNION <expressie> )
```

Voorbeeld 2.11: Geef de verschillende plaatsnamen die in het WOONPLAATS-attribuut van de WERKNEMER-relatie en in het LOKATIE-attribuut van de AFDELING-relatie staan.

```
(WOONPLAATS OF WERKNEMER) UNION (LOKATIE OF AFDELING)
```

Resultaat:

```
WOONPLAATS
----------
Amsterdam
Wateringen
Voorburg
Schiedam
Amstelveen
Rijswijk
Rotterdam
```

Toelichting: De twee project-operatoren leveren twee tussenresultaten op bestaande uit één attribuut. Het tussenresultaat van de eerste project-operator is:

```
WOONPLAATS
----------
Amsterdam
Wateringen
Voorburg
Schiedam
Voorburg
Amstelveen
Rijswijk
```

Het tussenresultaat van de tweede project-operator is:

```
LOKATIE
---------
Rotterdam
Amsterdam
Voorburg
```

Met de union-operator worden de twee tussenresultaten samengevoegd. Net als bij de project-operator worden dubbele rijen verwijderd. De plaatsnamen Amsterdam en Voorburg komen bijvoorbeeld in beide tussenresultaten voor, maar worden in het eindresultaat slechts éénmaal opgenomen. De namen van de attributen in het resultaat van een union-operator zijn gelijk aan de namen van de attributen van de eerst gespecificeerde relatie.

Relaties die met een union-operator gekoppeld worden, moeten aan twee voorwaarden voldoen:

- De twee relaties die 'ge-union-ed' worden, moeten dezelfde graad en dus een gelijk aantal attributen hebben.
- De attributen die 'ge-union-ed' worden (in voorbeeld 2.11 zijn dit WOONPLAATS en LOKATIE), moeten op hetzelfde domein gedefinieerd zijn.

Als twee relaties aan deze eisen voldoen zijn ze union-compatible. De onderstaande expressie is niet geldig, omdat de attributen ANR en WNR op verschillende domeinen gedefinieerd zijn. De twee tussenresultaten zijn dus niet union-compatible. In sommige situaties is zoiets dergelijks toch gewenst.

```
(WNR OF WERKNEMER) UNION (ANR OF AFDELING)
```

Voorbeeld 2.12: Geef de namen en de woonplaatsen van de werknemers die in Wateringen of Voorburg wonen.

```
(NAAM, WOONPLAATS OF
   ((WERKNEMER WHERE WOONPLAATS = 'Wateringen') UNION
    (WERKNEMER WHERE WOONPLAATS = 'Voorburg')))
```

Resultaat:

```
NAAM         WOONPLAATS
-----------  ----------
Homme        Wateringen
Hondijk      Voorburg
Osewoudt     Voorburg
```

2.8 Difference-operator: van elkaar aftrekken van relaties

Met de *difference-operator* (verschil) worden bepaalde rijen in een relatie geselecteerd die *niet* in een andere relatie voorkomen.

```
<difference-operator> ::= ( <expressie> MINUS <expressie> )
```

Voorbeeld 2.13: Geef de werknemers die niet in Voorburg wonen.

```
(WERKNEMER MINUS (WERKNEMER WHERE WOONPLAATS = 'Voorburg'))
```

Resultaat:

```
WNR  NAAM         ANR  SALARIS  WOONPLAATS  GEB-PLAATS
---  -----------  ---  -------  ----------  ----------
 10  Dingelam     104    70000  Amsterdam   Amsterdam
 15  Homme        102    60000  Wateringen  Rotterdam
 45  Alberegt     102    75000  Schiedam    Wassenaar
 80  Geyerstein   104    62000  Amstelveen  ?
 90  Hermans      108    55000  Rijswijk    Amsterdam
```

Toelichting: Het tussenresultaat van de select-operator bevat alle werknemers uit Voorburg. Het eindresultaat van de difference-operator bevat alle rijen uit de WERKNEMER-relatie, die *niet* in het tussenresultaat van de select-operator voorkomen, dus niet in Voorburg wonen.

Voorbeeld 2.14: Geef de plaatsnamen van de plaatsen waar wel werknemers wonen, maar waar geen afdelingen gevestigd zijn.

```
((WOONPLAATS OF WERKNEMER) MINUS (LOKATIE OF AFDELING))
```

Resultaat:

```
WOONPLAATS
----------
Wateringen
Schiedam
Amstelveen
Rijswijk
```

Voorbeeld 2.15: Van welke werknemers is de geboorteplaats onbekend?

```
(WERKNEMER MINUS (WERKNEMER WHERE GEB-PLAATS = GEB-PLAATS))
```

Resultaat:

```
WNR  NAAM         ANR  SALARIS  WOONPLAATS  GEB-PLAATS
---  -----------  ---  -------  ----------  ----------
 25  Hondijk      108    40000  Voorburg    ?
 80  Geyerstein   104    62000  Amstelveen  ?
```

Net als bij de union-operator geldt dat ten eerste de difference-operator alleen gebruikt mag worden bij twee union-compatible relaties, en dat ten tweede de namen van de attributen van het resultaat van de difference-operator gelijk zijn aan de namen van de eerst gespecificeerde relatie.

2.9 Semantic override

Voor alle operatoren geldt dat twee attributen alleen met elkaar vergeleken mogen worden als ze op hetzelfde domein gedefinieerd zijn. In sommige situaties is deze eis echter te 'streng'. Er bestaat daarom een mogelijkheid deze regel te negeren. Dit wordt *semantic override* genoemd. Semantic override kan gezien worden als toestaan dat appels met peren vergeleken worden.

Voorbeeld 2.16: Geef elke werknemer waarvan de naam gelijk is aan de naam van zijn of haar woonplaats. De attributen NAAM en WOONPLAATS zijn op verschillende domeinen gedefinieerd. Een directe vergelijking is dus niet toegestaan. Maar de volgende expressie is wel correct.

```
(WERKNEMER WHERE NAAM* = WOONPLAATS*)
```

Omdat in ons voorbeeld geen enkele werknemer een naam heeft die gelijk is aan de naam van zijn woonplaats, levert deze expressie een leeg resultaat op.

Toelichting: Door het plaatsen van een * achter de attributen die vergeleken moeten worden, wordt semantic override geforceerd. Een attribuut met een * wordt gezien als een attribuut met een universeel domein dat met elk ander attribuut met een universeel domein vergeleken mag worden. Het oorspronkelijke domein wordt dus in deze expressie buiten beschouwing gelaten.

Voorbeeld 2.17: Geef een lijst met namen van werknemers en leraren.

```
((NAAM* OF WERKNEMER) UNION (LERAAR* OF CURSUS))
```

Resultaat:

```
NAAM
-----------
Dingelam
Homme
Hondijk
Alberegt
Osewoudt
Geyerstein
Hermans
Palm
Prater
Cools
Veen
```

Hoofdstuk 3

De relationele algebra (vervolg)

3.1 Inleiding

Met de vijf elementaire operatoren die we in het vorige hoofdstuk behandeld hebben kan in principe elke vraag beantwoord worden. Echter, als alleen elementaire operatoren gebruikt worden, zijn sommige oplossingen zeer lastig en omslachtig te formuleren. De relationele algebra kent daarom een aantal afgeleide operatoren. Alle afgeleide operatoren kunnen gedefinieerd worden met één of meer elementaire operatoren. De in dit boek behandelde afgeleide operatoren zijn niet de enig mogelijke afgeleide operatoren. In principe is iedereen vrij om zelf afgeleide operatoren te definiëren. We beperken ons tot die operatoren die in de literatuur het meest voorkomen en exact gedefinieerd en algemeen geaccepteerd zijn. Dit zijn:

- De join-operator (diverse soorten)
- De intersection-operator
- De division-operator
- De maybe-select-operator
- De maybe-join-operator
- De outer-union-operator
- De outer-difference-operator
- De outer-intersection-operator
- De outer-join-operator

3.2 Join-operator

Met de join-operator wordt het cartesisch product van twee relaties gevormd en daaruit worden rijen geselecteerd. De join-operator is in feite een combinatie van een product- en een select-operator.

Voorbeeld 3.1: Geef van elke werknemer het werknemersnummer en de naam van de afdeling waarvoor hij of zij werkt (zie voorbeeld 2.9).

Zonder de join-operator ziet de expressie er als volgt uit:

```
(WERKNEMER.WNR, AFDELING.NAAM OF
   (WERKNEMER TIMES AFDELING) WHERE
      WERKNEMER.ANR = AFDELING.ANR))
```

Resultaat:

```
WNR  NAAM
---  --------
 10  Verkoop
 15  Verkoop
 25  Magazijn
 45  Verkoop
 70  Magazijn
 80  Inkoop
 90  Magazijn
```

Dezelfde vraag kan ook (eenvoudiger) met de join-operator herschreven worden:

```
(WERKNEMER.WNR, AFDELING.NAAM OF
   JOIN (WERKNEMER [ANR], AFDELING [ANR]))
```

Toelichting: De twee ANR-attributen worden de *join-attributen* genoemd. Als de join-attributen gelijke namen hebben en als ze de enige attributen in de betreffende relaties zijn met gelijke namen, mogen ze weggelaten worden. De expressie is dus gelijkwaardig aan de volgende:

```
(WERKNEMER.WNR, AFDELING.NAAM OF
   JOIN (WERKNEMER, AFDELING))
```

Deze vorm van een expressie noemen we de *impliciete vorm* van de join, in tegenstelling tot de *expliciete vorm*.

Voorbeeld 3.2: Geef het nummer van elke werknemer uit Voorburg, gevolgd door de naam van zijn of haar afdeling.

```
(WERKNEMER.WNR, AFDELING.NAAM OF
   (JOIN (WERKNEMER [ANR], AFDELING [ANR])
      WHERE WOONPLAATS = 'Voorburg'))
```

Of kortweg:

```
(WERKNEMER.WNR, AFDELING.NAAM OF
   (JOIN (WERKNEMER, AFDELING)
      WHERE WOONPLAATS = 'Voorburg'))
```

Resultaat:

```
WNR  NAAM
---  -----------
 25  Magazijn
 70  Magazijn
```

Voorbeeld 3.3: Geef het nummer en de naam van elke werknemer, gevolgd door de naam van zijn of haar baas.

```
SYNONYM B : WERKNEMER

(WERKNEMER.WNR, WERKNEMER.NAAM, B.NAAM OF
   JOIN (JOIN (WERKNEMER [ANR] AFDELING [ANR])
        [AFDELING.BAAS], B [WNR]))
```

Resultaat:

```
WNR  NAAM         NAAM
---  -----------  -----------
 10  Dingelam     Dingelam
 15  Homme        Alberegt
 25  Hondijk      Hermans
 45  Alberegt     Alberegt
 70  Osewoudt     Hermans
 80  Geyerstein   Dingelam
 90  Hermans      Hermans
```

Laten we eens even stilstaan bij de join-operator zelf. Waar staat de expressie `JOIN (WERKNEMER, AFDELING)` nu precies voor? Deze expressie is gelijkwaardig aan de volgende expressie:

```
((WERKNEMER TIMES AFDELING)
   WHERE WERKNEMER.ANR = AFDELING.ANR)
```

Wat opvalt is dat de twee join-attributen met een gelijkteken vergeleken worden. Deze join noemen we daarom ook *equi-join*. De voorbeelden 3.1 tot en met 3.3 zijn alle drie voorbeelden waarbij een equi-join-operator gebruikt is. Als we in dit boek de join-operator gebruiken, bedoelen we daarmee de equi-join-operator.

Er bestaan ook andere vormen van join-operatoren. We geven een voorbeeld.

Voorbeeld 3.4: Geef de naam en de tijdsduur van de cursus met de langste tijdsduur.

```
SYNONYM CUR2 : CURSUS

(CURSUS.NAAM, CURSUS.DUUR OF
   GT-JOIN (CURSUS [DUUR], CUR2 [DUUR]))
```

Resultaat:

```
NAAM         DUUR
-----------  ----
Statistiek     10
```

Dit is een voorbeeld van een *greater-than-join-operator*. Bij deze join worden de join-attributen met het groter-dan symbool vergeleken. Hierbij moet dan de waarde van het eerste join-attribuut groter zijn dan die van het tweede. De bovenstaande expressie is dus gelijk aan:

```
SYNONYM CUR2 : CURSUS

(CURSUS.NAAM, CURSUS.DUUR OF
   ((CURSUS TIMES CUR2) WHERE CURSUS.DUUR > CUR2.DUUR))
```

De algebra kent uiteraard ook de volgende join-operatoren:

- LT-JOIN: less-than-join
- GE-JOIN: greater-or-equal-join
- LE-JOIN: less-or-equal-join
- NE-JOIN: not-equal-join

Een andere speciale vorm van de join-operator is de *natural-join-operator*. Een natural-join is een equi-join, waarbij slechts één van van de twee join-attributen wordt opgenomen.

Voorbeeld 3.5: Geef de natural-join van de DIPLOMA- met de CURSUS-relatie.

```
NATURAL-JOIN (DIPLOMA, CURSUS)
```

Resultaat:

```
CNR  WNR  DATUM   NAAM         DUUR  LERAAR
---  ---  ------  -----------  ----  ------
 10   15  851110  Verkopen        2  Palm
 10   45  851104  Verkopen        2  Palm
 30   10  861220  Organiseren     4  Cools
 30   15  850921  Organiseren     4  Cools
 30   45  850422  Organiseren     4  Cools
 40   90  860514  Statistiek     10  Veen
 20   15  860630  Presenteren     4  Prater
 40   15  850731  Statistiek     10  Veen
```

De bovenstaande expressie is dus gelijk aan:

```
(DIPLOMA.CNR, WNR, DATUM, NAAM, DUUR, LERAAR OF
   ((DIPLOMA TIMES CURSUS) WHERE DIPLOMA.CNR = CURSUS.CNR))
```

Twee dingen vallen op. Ten eerste, de twee join-attributen worden met een gelijkteken vergeleken. En ten tweede, van de twee join-attributen wordt alleen CNR uit de DIPLOMA-relatie weergegeven. Het andere join-attribuut wordt weggelaten.

Een join heeft een onverwacht resultaat als één van of beide join-attributen een NULL-waarde bevat(ten). We zullen dit met een voorbeeld illustreren. Hierbij maken we gebruik van de twee onderstaande relaties.

```
De LEVERANCIER-relatie:      De KLANT-relatie:

LNAAM      STAD              KNAAM      STAD
---------  ---------         ---------  ---------
Hogenboom  Voorburg          Cools      ?
Shell      ?                 Leeuwen    Eindhoven
IBM        Voorburg          Janssens   Voorburg
```

Voorbeeld 3.6: Geef de gegevens over leveranciers die in een stad gevestigd zijn waar ook klanten wonen.

```
JOIN (LEVERANCIER, KLANT)
```

Resultaat:

```
LNAAM      STAD       KNAAM      STAD
---------  ---------  ---------  ---------
Hogenboom  Voorburg   Janssens   Voorburg
IBM        Voorburg   Janssens   Voorburg
```

Hoe ontstaat dit resultaat? Het ontstaan van dit resultaat wordt duidelijk als eerst het product van deze twee relaties wordt getoond:

```
LNAAM      STAD       KNAAM      STAD
---------  ---------  ---------  ---------
Hogenboom  Voorburg   Cools      ?
Hogenboom  Voorburg   Leeuwen    Eindhoven
Hogenboom  Voorburg   Janssens   Voorburg
Shell      ?          Cools      ?
Shell      ?          Leeuwen    Eindhoven
Shell      ?          Janssens   Voorburg
IBM        Voorburg   Cools      ?
IBM        Voorburg   Leeuwen    Eindhoven
IBM        Voorburg   Janssens   Voorburg
```

Uit dit resultaat worden nu alleen die rijen geselecteerd waar de waarden van de join-attributen gelijk zijn; dit zijn er slechts twee (de derde en de laatste rij). In de andere rijen zijn de waarden of ongelijk, of één van de

waarden is NULL, of beide waarden zijn NULL. En zoals reeds uitgelegd, NULL-waarden voldoen nooit aan een conditie.

Het resultaat van de volgende expressie bestaat (toevallig) ook uit slechts twee rijen.

```
NE-JOIN (LEVERANCIER, KLANT)
```

Resultaat:

```
LNAAM      STAD       KNAAM      STAD
---------  ---------  ---------  ---------
Hogenboom  Voorburg   Leeuwen    Eindhoven
IBM        Voorburg   Leeuwen    Eindhoven
```

Conclusie: Het resultaat van de volgende twee expressies is dus alleen gelijk als de join-attributen *geen* NULL-waarden bevatten.

```
((JOIN (LEVERANCIER, KLANT)) UNION
(NE-JOIN (LEVERANCIER, KLANT)))
```

en

```
(LEVERANCIER TIMES KLANT)
```

3.3 Intersection-operator: doorsnede van relaties

Met de *intersection-operator* worden rijen geselecteerd die in twee relaties voorkomen.

```
<intersection-operator> ::= ( <expressie> INTERSECT <expressie> )
```

Voorbeeld 3.7: Geef elke werknemer die in Amsterdam geboren is en meer dan € 60.000,- verdient.

```
((WERKNEMER WHERE SALARIS > 60000) INTERSECT
 (WERKNEMER WHERE GEB-PLAATS = 'Amsterdam'))
```

Resultaat:

```
WNR  NAAM        ANR  SALARIS  WOONPLAATS  GEB-PLAATS
---  ----------  ---  -------  ----------  ----------
 10  Dingelam    104    70000  Amsterdam   Amsterdam
```

Toelichting: Met de eerste select-operator worden alle werknemers geselecteerd die meer dan € 60.000,- verdienen. Met de tweede select-operator worden alle werknemers geselecteerd die in Amsterdam wonen. De intersection-operator zoekt alle rijen op die in beide (tussen)resultaten voorkomen, dus aan beide condities voldoen.

De bovenstaande expressie kan ook met alleen een select-operator geformuleerd worden en zonder intersection-operator (ga dit zelf na). Bij het volgende voorbeeld is dit echter niet mogelijk.

Voorbeeld 3.8: Geef de nummers van de werknemers die van geen enkele afdeling baas zijn.

```
(WNR OF WERKNEMER) INTERSECT (BAAS OF AFDELING)
```

Resultaat:

```
WNR
---
 15
 25
 70
 80
```

3.4 Division-operator: delen van relaties

We beginnen met het behandelen van een eenvoudig voorbeeld van de *division-operator*. Veronderstel dat we de volgende twee relaties hebben (let op, deze twee relaties wijken af van het oorspronkelijke voorbeeld):

```
De CURSUS-relatie:

CNR
---
 10
 20
```

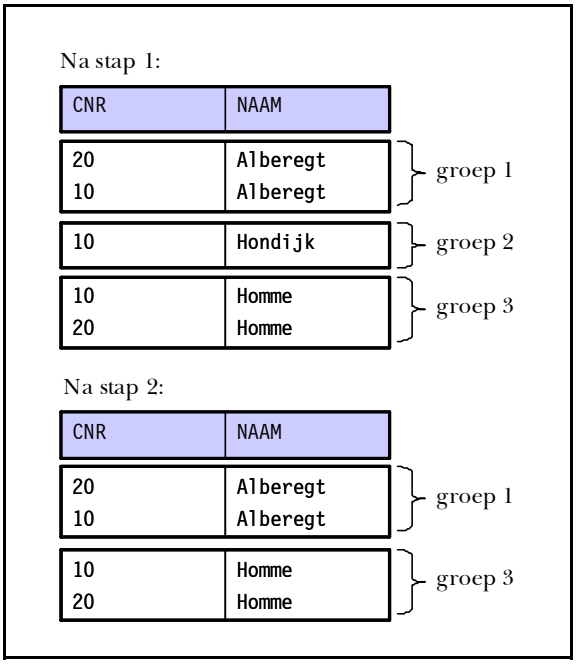
```
De DIPLOMA-relatie:

CNR  NAAM
---  --------
 10  Homme
 10  Hondijk
 20  Alberegt
 10  Alberegt
 20  Homme
```

Voorbeeld 3.9: Geef de naam van elke werknemer die tenminste alle cursussen heeft gevolgd die in de CURSUS-relatie geregistreerd staan.

```
(NAAM OF (DIPLOMA [CNR] DIVIDEBY (CNR OF CURSUS)))
```

Resultaat:

```
NAAM
--------
Homme
Alberegt
```



Figuur 3.3 *Voorbeeld van de division-operator*

Wat is er nu gebeurd? In het kort komt het er op neer dat de DIPLOMA-relatie 'gedeeld' is door de CURSUS-relatie op basis van de CNR-attributen. De bewerking van de division-operator kan men als volgt in drie stappen beschrijven (zie tevens figuur 3.3):

Stap 1: De rijen in de DIPLOMA-relatie worden gegroepeerd. Als alle waarden in een verzameling rijen in elke attribuut *behalve* het CNR-attribuut overeenkomen, dan vormen zij een groep.

Stap 2: Per groep wordt nu gekeken of de verzameling CNR-waarden in de CURSUS-relatie een deelverzameling is van de verzameling CNR-waarden in de groep. De groepen die hieraan voldoen worden geselecteerd. In het voorbeeld zijn dit de groepen 1 en 3.

Stap 3: Het eindresultaat bestaat uit alle attributen van dit tussenresultaat minus het CNR-attribuut en bestaat uit een aantal rijen dat overeenkomt met het aantal groepen. Het eindresultaat is hiervoor weergegeven.

Opmerking: Als we het eindresultaat met de oorspronkelijke CURSUS-relatie vermenigvuldigen, ontstaat een resultaat dat een deelverzameling is van de oorspronkelijke DIPLOMA-relatie (we noemen het eindresultaat voor het gemak de RESULTAAT-relatie):

```
(CURSUS TIMES RESULTAAT)
```

Resultaat:

```
CNR  NAAM
---  --------
 20  Alberegt
 10  Alberegt
 10  Homme
 20  Homme
```

De rijen uit de DIPLOMA-relatie die *niet* in het bovenstaande resultaat voorkomen vormen *restrijen* van de division-operator. De rest van een division-operator is te vergelijken met de rest van een normale deling van een getal door een ander getal.

De division-operator kent net als de join-operator een impliciete vorm. Als de naam van het attribuut dat gedeeld wordt gelijk is aan de naam van het attribuut dat deelt en er niet meerdere gelijk zijn, dan mogen die namen weggelaten worden. Dus de expressie wordt:

```
(NAAM OF (DIPLOMA DIVIDEBY CURSUS))
```

We zullen nog een voorbeeld geven, waarbij we uitgaan van het standaardvoorbeeld.

Voorbeeld 3.10: Geef de nummers van de werknemers die alle cursussen gevolgd hebben die door Cools gedoceerd worden.

```
(WNR OF (DIPLOMA[CNR] DIVIDEBY
        (CNR OF (CURSUS WHERE LERAAR = 'Cools'))))
```

Impliciete vorm:

```
(WNR OF (DIPLOMA DIVIDEBY (CURSUS WHERE LERAAR = 'Cools')))
```

Resultaat:

```
WNR
---
 10
 15
 45
```

3.5 Maybe-select-operator: selecteren op NULL-waarden

Met de *maybe-select-operator* worden alle rijen geselecteerd waarvan de waarde in een bepaald attribuut gelijk is aan de *NULL-waarde*.

```
<maybe-select-operator> ::= ( <expressie> WHERE <attribuut> IS NULL )
```

Voorbeeld 3.11: Geef de werknemers waarvan de geboorteplaats onbekend is.

```
(WERKNEMER MINUS (WERKNEMER WHERE GEB-PLAATS = GEB-PLAATS))
```

Resultaat:

```
WNR  NAAM         ANR  SALARIS  WOONPLAATS  GEB-PLAATS
---  -----------  ---  -------  ----------  ----------
 25  Hondijk      108    40000  Voorburg    ?
 80  Geyerstein   104    62000  Amstelveen  ?
```

Toelichting: Het eindresultaat bevat elke rij uit de WERKNEMER-relatie waar het GEB-PLAATS-attribuut gelijk is aan de NULL-waarde. Met de maybe-select-operator kan de vraag als volgt geformuleerd worden:

```
(WERKNEMER WHERE GEB-PLAATS IS NULL)
```

3.6 Maybe-join-operator

Met de *maybe-join-operator* worden rijen in een relatie geselecteerd waarvan de waarde in één van de twee join-attributen of van beide join-attributen gelijk is aan de NULL-waarde.

In paragraaf 3.2 bleek reeds dat een rij niet in het resultaat van een join-operator voorkomt als één van de join-attributen geen waarde heeft die overeenkomt met een waarde in één van de rijen van de andere relatie. We gebruiken nogmaals de LEVERANCIER- en KLANT-relaties uit voorbeeld 3.6. De volgende maybe-join levert nu het onderstaande resultaat op:

```
MAYBE-JOIN (LEVERANCIER [STAD], KLANT [STAD])
```

Impliciete vorm:

```
MAYBE-JOIN (LEVERANCIER, KLANT)
```

Resultaat:

```
LNAAM      STAD       KNAAM      STAD
---------  ---------  ---------  ---------
Hogenboom  Voorburg   Cools      ?
Shell      ?          Cools      ?
Shell      ?          Leeuwen    Eindhoven
Shell      ?          Janssens   Voorburg
IBM        Voorburg   Cools      ?
```

Het resultaat bevat alle rijen van het product van de twee relaties waar minimaal één van de twee join-attributen de NULL-waarde bevat.

3.7 De outer-versie van vier operatoren

De operatoren union, difference en intersection vereisen dat de twee betrokken expressies union-compatible zijn. Van elk van deze drie operatoren bestaat echter ook een versie waarvoor deze eis niet geldt. Deze operatoren worden respectievelijk outer-union, outer-difference en outer-intersection genoemd. Voor de join-operator bestaat er ook een outer-versie: de outer-join-operator.

3.7.1 Outer-union-operator

Met een *outer-union-operator* worden relaties met één of meer overeenkomstige attributen onder elkaar geplakt. Om de outer-union-operator toe te lichten gebruiken we de twee onderstaande relaties. De SECRETARESSE-relatie bevat van elke secretaresse het werknemersnummer en de tiksnelheid. De PROGRAMMEUR-relatie bevat van elke programmeur het werknemersnummer, de programmeertaal waarin hij of zij gespecialiseerd is en ook de tiksnelheid. Alle secretaressen en programmeurs vormen de verzameling van alle werknemers.

De SECRETARESSE-relatie:

WNR	SNELHEID
150	270
160	250
200	150

De PROGRAMMEUR-relatie:

WNR	PROG-TAAL	SNELHEID
200	Cobol	150
210	Pascal	140
220	Fortran	190

Voorbeeld 3.12: Geef van elke werknemer of de programmeertaal waarin hij of zij gespecialiseerd is of zijn of haar tiksnelheid.

```
(SECRETARESSE [WNR] OUTER-UNION PROGRAMMEUR [WNR])
```

Resultaat:

WNR	SNELHEID	PROG-TAAL	SNELHEID
150	270	?	?
160	250	?	?
200	150	Cobol	150
210	?	Pascal	140
220	?	Fortran	190

Toelichting: De WNR-attributen van beide relaties worden onder elkaar geplaatst en de attributen die niet gespecificeerd zijn staan ernaast. Dit betekent dat een rij aangevuld wordt met NULL-waarden voor die attributen waarvoor geen waarde bekend is. Voor alle secretaresses is de programmeertaal onbekend en wordt deze kolom met NULL-waarden gevuld. Voor elke programmeur wordt de tiksnelheid met een NULL-waarde aangegeven.

Voorbeeld 3.13: Geef van elke werknemer de programmeertaal waarin hij of zij gespecialiseerd is en zijn of haar tiksnelheid.

```
(SECRETARESSE [WNR, SNELHEID] OUTER-UNION
 PROGRAMMEUR [WNR, SNELHEID])
```

Of kortweg (impliciete vorm):

```
(SECRETARESSE OUTER-UNION PROGRAMMEUR)
```

Resultaat:

```
WNR  SNELHEID  PROG-TAAL
---  --------  ---------
150       270  ?
160       250  ?
200       150  Cobol
210       140  Pascal
220       190  Fortran
```

3.7.2 Outer-difference-operator

Voorbeeld 3.14: Geef het nummer en de tiksnelheid van elke secretaresse die *geen* programmeur is.

```
(SECRETARESSE [WNR, SNELHEID] OUTER-MINUS
 PROGRAMMEUR  [WNR, SNELHEID])
```

Impliciete vorm:

```
(SECRETARESSE OUTER-MINUS PROGRAMMEUR)
```

Resultaat:

```
WNR  SNELHEID
---  --------
150       270
160       250
```

Toelichting: Alleen secretaresses waarvan het werknemersnummer in de PROGRAMMEUR-relatie voorkomt, worden in het resultaat opgenomen. Voor elke secretaresse die aan de conditie voldoet, worden alle attributen weergegeven.

3.7.3 Outer-intersection-operator

Voorbeeld 3.15: Geef van elke secretaresse die tevens programmeur is het nummer, de tiksnelheid en de programmeertaal.

```
(SECRETARESSE [WNR] OUTER-INTERSECT PROGRAMMEUR [WNR])
```

Resultaat:

```
WNR  SNELHEID  PROG-TAAL
---  --------  ---------
200       150  Cobol
```

Toelichting: Van elk werknemersnummer dat in beide relaties voorkomt wordt de SNELHEID en de PROG-TAAL weergegeven.

3.7.4 Outer-join-operator

Voorbeeld 3.16: Geef van elke werknemer het nummer, de naam en de afdeling die hij of zij leidt. De 'normale' join-operator geeft het volgende resultaat:

```
(WERKNEMER.WNR, WERKNEMER.NAAM, AFDELING.NAAM OF
   (JOIN (AFDELING, WERKNEMER)))
```

Resultaat:

```
WNR  NAAM      NAAM
---  --------  --------
 45  Alberegt  Verkoop
 10  Dingelam  Inkoop
 90  Hermans   Magazijn
```

Alleen van die werknemers die een afdeling leiden wordt de naam en de afdelingsnaam geselecteerd. Veronderstel nu dat het resultaat ten eerste letterlijk *elke* werknemer moet bevatten. Veronderstel vervolgens dat als een werknemer geen baas is van een afdeling de NULL-waarde afgedrukt moet worden. Dit resultaat is met de volgende expressie te verkrijgen.

```
(WERKNEMER.WNR, WERKNEMER.NAAM, AFDELING.NAAM OF
   (JOIN (WERKNEMER, AFDELING)) OUTER-UNION
(WERKNEMER.WNR, WERKNEMER.NAAM OF
   (WERKNEMER OUTER-MINUS AFDELING))
```

Resultaat:

```
WNR  NAAM         NAAM
---  -----------  --------
 10  Dingelam     Inkoop
 45  Alberegt     Verkoop
 90  Hermans      Magazijn
 15  Homme        ?
 25  Hondijk      ?
 70  Osewoudt     ?
 80  Geyerstein   ?
```

Deze expressie is met behulp van een *outer-join-operator* eenvoudiger te formuleren.

```
(WERKNEMER.WNR, WERKNEMER.NAAM, AFDELING.NAAM OF
   (OUTER-JOIN (WERKNEMER, AFDELING)))
```

In feite is dit een *outer-equi-join-operator*

Voorbeeld 3.17: Geef de naam van elke werknemer plus de naam van de afdeling waarvan hij of zij de baas is. Als de werknemer van geen enkele afdeling baas is, geef dan de NULL-waarde.

```
(WERKNEMER.NAAM, AFDELING.NAAM OF
   (OUTER-JOIN (WERKNEMER [WNR], AFDELING [BAAS])))
```

Resultaat:

```
NAAM         NAAM
-----------  --------
Dingelam     Inkoop
Homme        ?
Hondijk      ?
Alberegt     Verkoop
Osewoudt     ?
Geyerstein   ?
Hermans      Magazijn
```

Laten we eens stap voor stap de verwerking van deze expressie toelichten. Ten eerste wordt het product van de WERKNEMER-relatie met de AFDELING-relatie berekend. Resultaat:

```
WNR  NAAM         ANR  ...  ANR  NAAM      BAAS  ...
---  -----------  ---  ---  ---  -------   ----  ---
 10  Dingelam     104  ...  102  Verkoop     45  ...
 10  Dingelam     104  ...  104  Inkoop      10  ...
 10  Dingelam     104  ...  108  Magazijn    90  ...
 15  Homme        102  ...  102  Verkoop     45  ...
 15  Homme        102  ...  104  Inkoop      10  ...
 15  Homme        102  ...  108  Magazijn    90  ...
 25  Hondijk      108  ...  102  Verkoop     45  ...
 25  Hondijk      108  ...  104  Inkoop      10  ...
 25  Hondijk      108  ...  108  Magazijn    90  ...
 45  Alberegt     102  ...  102  Verkoop     45  ...
 45  Alberegt     102  ...  104  Inkoop      10  ...
 45  Alberegt     102  ...  108  Magazijn    90  ...
  :  :              :        :  :            :
  :  :              :        :  :            :
 90  Hermans      108  ...  102  Verkoop     45  ...
 90  Hermans      108  ...  104  Inkoop      10  ...
 90  Hermans      108  ...  108  Magazijn    90  ...
```

Uit dit tussenresultaat worden alle rijen geselecteerd waarin de waarde van het WERKNEMER.WNR-attribuut gelijk is aan de waarde van het AFDELING.BAAS-attribuut. Resultaat:

```
WNR  NAAM         ANR  ...  ANR  NAAM      BAAS  ...
---  -----------  ---  ---  ---  -------   ----  ---
 10  Dingelam     104  ...  104  Inkoop      10  ...
 45  Alberegt     102  ...  102  Verkoop     45  ...
 90  Hermans      108  ...  108  Magazijn    90  ...
```

Uit dit tussenresultaat worden de attributen WERKNEMER.NAAM en AFDELING.NAAM geprojecteerd:

```
NAAM         NAAM
-----------  -------
Dingelam     Inkoop
Alberegt     Verkoop
Hermans      Magazijn
```

Tenslotte wordt aan dit resultaat ten eerste elke werknemer toegevoegd die geen baas is van een afdeling en ten tweede elke afdeling waarvoor geen baas bestaat (als er zo'n afdeling zou zijn).

3.7.5 Left- en right-outer-join-operator

Speciale varianten van de outer-join zijn de left- en right-outer-join. Met een *left-outer-join* worden alleen rijen aan het tussenresultaat van de join toegevoegd, die gebaseerd zijn op de relatie welke links in de expressie vermeld staat.

Voorbeeld 3.18: Wat is het resultaat van de volgende join?

```
LEFT-OUTER-JOIN (WERKNEMER, AFDELING))
```

Het resultaat bevat alle werknemers die in een afdeling werken en alle werknemers die niet voor een afdeling werken. Het resultaat bevat geen gegevens over afdelingen zonder werknemers (die wel toegevoegd zouden worden bij een 'normale' outer-join).

Voorbeeld 3.19: Wat is het resultaat van de volgende join?

```
RIGHT-OUTER-JOIN (WERKNEMER, AFDELING)
```

Het resultaat bevat alle werknemers die in een afdeling werken en alle afdelingen zonder werknemers, maar bevat geen werknemers die niet voor een afdeling werken.

3.8 Definities van de afgeleide operatoren

In deze paragraaf geven we aan hoe expressies met afgeleide operatoren omgezet kunnen worden naar expressies bestaande uit alleen elementaire operatoren. We gaan daarbij uit van het volgende:

- expressie E1 bestaat uit de attributen A1, A2, A3
- expressie E2 bestaat uit de attributen A3, A4, A5
- expressie E3 bestaat uit de attributen A1, A2, A3
- attribuut Ai (1≤i≤5) is op domein Di gedefinieerd
- θ is een vergelijkings-operator

```
JOIN (E1, E2)  ==>              ((E1 TIMES E2) WHERE E1.A3 = E2.A3)
GT-JOIN (E1, E2)  ==>           ((E1 TIMES E2) WHERE E1.A3 > E2.A3)
NATURAL-JOIN (E1, E2)  ==>      (A1, A2, E1.A3, A4, A5 OF
                                  ((E1 TIMES E2) WHERE E1.A3 = E2.A3))
(E1 INTERSECT E3)  ==>          (E1 MINUS (E1 MINUS E3))
                                of:
                                (E3 MINUS (E3 MINUS E1))
(E1 DIVIDEBY E2)  ==>           ((A1, A2 OF E1) MINUS
                                 (A1, A2 OF ((A1, A2, A3 OF
                                    ((A1, A2 OF E1) TIMES E2)) MINUS E1)))
(E1 WHERE A1 IS NULL)  ==>      (E1 MINUS (E1 WHERE A1 = A1))
MAYBE-JOIN (E1, E2)  ==>        (E1 TIMES E2) MINUS
                                   (((E1 TIMES E2) WHERE E1.A3 = E2.A3) UNION
                                    ((E1 TIMES E2) WHERE E1.A3 <> E2.A3))
(E1 OUTER-UNION E2)  ==>        ((E1 TIMES (<D4:?, D5:?>) UNION
                                 (<D1:?, D2:?>) TIMES E2))
(E1 OUTER-MINUS E2)  ==>        (E1 MINUS ((A1, A2, E1.A3 OF
                                    ((E1 TIMES E2) WHERE E1.A3 = E2.A3)))
(E1 OUTER-INTERSECT E2)  ==>    (A1, A2, E1.A3, A4, A5 OF
                                   ((E1 TIMES E2) WHERE E1.A3 = E2.A3))
```

Voor de outer-join-operator definiëren we eerst twee andere relaties:

```
      R1 := (E1 MINUS (A1, A2, E1.A3 OF (JOIN(E1, E2)))
      R2 := (E2 MINUS (E2.A3, A4, A5 OF (JOIN(E1, E2)))

OUTER-JOIN (E1, E2)  ==>        ((E1 TIMES E2) WHERE E1.A3 = E2.A3) UNION
                                (R1 TIMES <D4:?, D5,?>) UNION
                                (<D1:?, D2:?> TIMES R2)
```

Voor de left-outer-join-operator gebruiken we de (bovenstaande) relatie R1 weer.

```
LEFT-OUTER-JOIN (E1, E2)  ==>   ((E1 TIMES E2) WHERE E1.A3 = E2.A3) UNION
                                (R1 TIMES <D4:?, D5,?>)
```

Hoofdstuk 4

De relationele calculus

4.1 Inleiding

De *relationele calculus* is een bepaalde stijl voor het formuleren van expressies. Zij is gebaseerd op de *predikatenlogica*. De relationele calculus kent geen operatoren zoals select en project, maar *formules* die waar, niet waar of onbekend zijn. E.F. Codd introduceerde de relationele calculus[3] in 1972; zie [CODD72]. Van de relationele calculus bestaan twee stijlen: de *relationele rij-calculus* en de *relationele domein-calculus*. In dit boek wordt alleen de eerstgenoemde stijl behandeld. Omdat zeer weinig databasetalen op de relationele domein-calculus gebaseerd zijn, laten we deze buiten beschouwing. We verwijzen hiervoor naar [DATE86] en [ULLM82]. We gebruiken in dit boek de term relationele calculus en bedoelen daar dan de relationele rij-calculus mee.

Voorbeelden van databasetalen die op de relationele calculus zijn gebaseerd, zijn: SQL, INGRES/QUEL van Ingres en Query-By-Example van IBM.

4.2 De rij-variabele

In calculus-expressies spelen rij-variabelen een centrale rol. Een *rij-variabele* (tuple variable) in een calculus-expressie is te vergelijken met een variabele in een programmeertaal. We zullen verder in plaats van rij-variabele kortweg de term *variabele* gebruiken. Voordat een variabele in een expressie gebruikt kan worden, moet eerst het *bereik* van de variabele aangegeven worden. Met het bereik wordt aangegeven welke *mogelijke waarden* een variabele kan aannemen. Het bereik wordt door middel van een opsomming van expressies aangegeven.

```
<rij-variabele-definitie> ::= VAR <rij-variabelenaam> : <bereik> { , <bereik> }

<bereik> ::= <relatienaam> | <expressie>
```

Voorbeeld van een expressie bestaande uit slechts de naam van één relatie:

```
VAR W : WERKNEMER
```

[3] We spreken wel van relationele *calculus*, maar eigenlijk moeten we spreken van relationele *logica*, want dit is de correcte vertaling. Echter, voor een goede aansluiting met de Engelstalige literatuur is gekozen voor 'relationele calculus'.

Dit betekent dat de variabele W alleen die waarden kan aannemen die overeenkomen met rijen in de WERKNEMER-relatie. De W-variabele kan dus in dit geval slechts één van de volgende zeven gedaantes aannemen:

```
<10, 'Dingelam'  , 104, 70000, 'Amsterdam' , 'Amsterdam'>
<15, 'Homme'     , 102, 60000, 'Wateringen', 'Rotterdam'>
<25, 'Hondijk'   , 108, 40000, 'Voorburg'  , ?>
<45, 'Alberegt'  , 102, 75000, 'Schiedam'  , 'Wassenaar'>
<70, 'Osewoudt'  , 108, 42000, 'Voorburg'  , 'Rijswijk'>
<80, 'Geyerstein', 104, 62000, 'Amstelveen', ?>
<90, 'Hermans'   , 108, 55000, 'Rijswijk'  , 'Amsterdam'>
```

In het bovenstaande voorbeeld is het bereik van de variabele beperkt tot één relatie. Maar zoals de definitie laat zien, mogen er ook meerdere relaties gebruikt worden. Ter illustratie introduceren we de volgende twee relaties die allebei hetzelfde relatieschema als de WERKNEMER-relatie hebben (zie paragraaf 1.3):

```
RELATIONTYPE WERKNEMER
   SCHEMA ( WNR         WNR    NOT NULL,
            NAAM        WNAAM  NOT NULL,
            ANR         ANR    ,
            SALARIS     GELD   NOT NULL,
            WOONPLAATS  PLAATS NOT NULL,
            GEB-PLAATS  PLAATS )

RELATION ANALIST
   RELATIONTYPE ( WERKNEMER )
   PRIM ( WNR )

RELATION SECRETARESSE
   RELATIONTYPE ( WERKNEMER )
   PRIM ( WNR )
```

De waarden van deze twee relaties zijn achtereenvolgens:

```
De ANALIST-relatie:

WNR  NAAM         ANR  SALARIS  WOONPLAATS  GEB-PLAATS
---  -----------  ---  -------  ----------  ----------
100  Speksnijder  104    90000  Rotterdam   Rotterdam
110  Janssens     102    85000  Schiedam    Rotterdam
120  Cools        108    70000  Leiden      ?

De SECRETARESSE-relatie:

WNR  NAAM         ANR  SALARIS  WOONPLAATS  GEB-PLAATS
---  -----------  ---  -------  ----------  ----------
120  Cools        108    70000  Leiden      ?
130  Bronwasser   108    62000  Voorburg    Leiden
```

Na de volgende specificatie van de AS-variabele:

```
VAR AS : ANALIST, SECRETARESSE
```

kan zij één van de volgende vier waarden hebben:

```
<100, 'Speksnijder', 104, 90000, 'Rotterdam', 'Rotterdam'>
<110, 'Janssens'   , 102, 85000, 'Schiedam' , 'Rotterdam'>
<120, 'Cools'      , 108, 70000, 'Leiden'   , ?>
<130, 'Bronwasser' , 108, 62000, 'Voorburg' , 'Leiden'>
```

Alle relaties in de definitie van een variabele moeten (uiteraard) union-compatible zijn (zie paragraaf 2.7).
Zoals uit de definitie blijkt, kan een rij-variabele ook met een calculus-expressie gedefinieerd worden. Het bespreken van deze vorm stellen we uit tot de expressie zelf volledig is behandeld.

4.3 De calculus-expressie

Een calculus-expressie is een beschrijving van de eisen waaraan een rij behoort te voldoen om tot het resultaat te mogen behoren. Dat ook de waarde van een calculus-expressie, net als een algebraïsche expressie, een relatie is, is niet zo verwonderlijk. Het is trouwens de enige overeenkomst die deze expressie heeft met de algebraïsche expressie. Een calculus-expressie kent geen operatoren zoals select en project, maar formules die waar kunnen zijn (enigszins te vergelijken met condities in de relationele algebra).

In deze en de volgende paragrafen wordt veelvuldig uitgegaan van de volgende variabelen:

```
VAR W  : WERKNEMER
VAR W2 : WERKNEMER
VAR A  : AFDELING
VAR A2 : AFDELING
VAR C  : CURSUS
VAR C2 : CURSUS
VAR D  : DIPLOMA
VAR D2 : DIPLOMA
```

Formules laten we in deze paragraaf nog buiten beschouwing. In de volgende paragrafen wordt hier uitvoerig op ingegaan. We beginnen met een beschrijving van de meest eenvoudige vorm van de calculus-expressie.

```
<expressie> ::=
   ( <attribuut> { , <attribuut> } ) |
   ( <attribuut> { , <attribuut> } WHERE <formule> )

<attribuut> :=
   <rij-variabelenaam>.<attribuutnaam> |
   <rij-variabelenaam>.<attribuutnaam>*
```

Voorbeeld 4.1: Geef alle afdelingen.

```
(A.ANR, A.NAAM, A.BAAS, A.LOKATIE)
```

Resultaat:

```
ANR  NAAM      WNR  LOKATIE
---  --------  ---  ---------
102  Verkoop    45  Rotterdam
104  Inkoop     10  Amsterdam
108  Magazijn   90  Voorburg
```

Toelichting: De expressie moet als volgt gelezen worden: 'Geef van alle mogelijke waarden die de variabele A kan aannemen het ANR-, NAAM-, BAAS- en het LOKATIE-attribuut'. Dit zijn alle rijen in de AFDELING-relatie. (Net als in de vorige twee hoofdstukken geven we ook in dit hoofdstuk de resultaten van expressies in tabelvorm weer.)

Het resultaat van een calculus-expressie is altijd een deelverzameling van het bereik van de variabele die gebruikt wordt. Met deelverzameling bedoelen we een horizontale èn een verticale deelverzameling.

Voorbeeld 4.2: Geef de nummers en namen van alle werknemers.

```
(W.WNR, W.NAAM)
```

Resultaat:

```
WNR  NAAM
---  -----------
 10  Dingelam
 15  Homme
 25  Hondijk
 45  Alberegt
 70  Osewoudt
 80  Geyerstein
 90  Hermans
```

Toelichting: De expressie moet als volgt gelezen worden: 'Geef de WNR- en NAAM-attributen van alle mogelijke waarden die de variabele W kan aannemen'. Deze expressie is te vergelijken met een project-operator in de relationele algebra. Het resultaat van dit voorbeeld is een verticale deelverzameling van het bereik van de W-variabele.

Uiteraard worden dubbele rijen uit het resultaat van een calculus-expressie verwijderd. We laten dat in het volgende voorbeeld zien.

Voorbeeld 4.3: Geef de namen van de plaatsen waar werknemers wonen.

```
(W.WOONPLAATS)
```

Resultaat:

```
WOONPLAATS
----------
Amsterdam
Wateringen
Voorburg
Schiedam
Amstelveen
Rijswijk
```

Voorbeeld 4.4: Geef het product van de WERKNEMER-relatie met de AFDELING-relatie.

```
(W.WNR, W.NAAM, ..., A.ANR, A.NAAM, A.WNR, A.LOKATIE)
```

Resultaat:

```
WNR  NAAM         ...  ANR  NAAM      BAAS  LOKATIE
---  -----------  ---  ---  --------  ----  ---------
 10  Dingelam     ...  102  Verkoop     45  Rotterdam
 15  Homme        ...  102  Verkoop     45  Rotterdam
 25  Hondijk      ...  102  Verkoop     45  Rotterdam
 45  Alberegt     ...  102  Verkoop     45  Rotterdam
 70  Osewoudt     ...  102  Verkoop     45  Rotterdam
 80  Geyerstein   ...  102  Verkoop     45  Rotterdam
 90  Hermans      ...  102  Verkoop     45  Rotterdam
 10  Dingelam     ...  104  Inkoop      10  Amsterdam
 15  Homme        ...  104  Inkoop      10  Amsterdam
 25  Hondijk      ...  104  Inkoop      10  Amsterdam
 45  Alberegt     ...  104  Inkoop      10  Amsterdam
 70  Osewoudt     ...  104  Inkoop      10  Amsterdam
 80  Geyerstein   ...  104  Inkoop      10  Amsterdam
 90  Hermans      ...  104  Inkoop      10  Amsterdam
 10  Dingelam     ...  108  Magazijn    90  Voorburg
 15  Homme        ...  108  Magazijn    90  Voorburg
 25  Hondijk      ...  108  Magazijn    90  Voorburg
 45  Alberegt     ...  108  Magazijn    90  Voorburg
 70  Osewoudt     ...  108  Magazijn    90  Voorburg
 80  Geyerstein   ...  108  Magazijn    90  Voorburg
 90  Hermans      ...  108  Magazijn    90  Voorburg
```

Toelichting: In de expressie worden twee variabelen gebruikt: W en A. Als in een expressie meer dan één variabele wordt gespecificeerd, wordt het product gecreëerd uit de mogelijke waarden die de betreffende variabelen kunnen aannemen. Het resultaat van de bovenstaande calculus-expressie is dus gelijk aan het cartesisch product van de twee relaties. Wederom geldt dan weer dat de waarde van de calculus-expressie een deelverzameling is van het product van de gebruikte variabelen.

Als een vergelijking gemaakt wordt tussen de relationele calculus en de relationele algebra, dan geldt het volgende: Indien W en A twee variabelen zijn, gedefinieerd op de relaties WERKNEMER (met attributen W1, W2, ..., Wn) en AFDELING (met attributen A1, A2, ..., An), dan is de calculus-expressie

```
(W.W1, W.W2, A.A1, A.A2)
```

gelijkwaardig aan de volgende algebraïsche expressie:

```
(W1, W2, A1, A2 OF (W TIMES A))
```

4.4 Formules

Met de *formule* in een calculus-expressie wordt aangegeven welke rijen geselecteerd moeten worden. Een formule heeft voor een bepaalde rij de waarde waar, onwaar of onbekend. Als de formule voor een bepaalde rij waar is, wordt zij in het resultaat opgenomen, in de andere twee gevallen niet. Een formule kan zeven verschillende vormen hebben, zie de definitie hieronder. In de volgende paragrafen zullen we deze vormen één voor één behandelen.

```
<formule> ::=
   <conditie> |
   <formule> <boolean-operator> <formule> |
   NOT ( <formule> ) |
   ( <formule> ) |
   EXISTS <rij-variabelenaam> ( <formule> ) |
   FORALL <rij-variabelenaam> ( <formule> ) |
   IF <formule> THEN <formule>

<boolean-operator> ::= AND | OR

<conditie> :=
   <attribuut> <vergelijkings-operator> <waarde> |
   <attribuut> <vergelijkings-operator> <attribuut>

<attribuut> :=
   <rij-variabelenaam> . <attribuutnaam> |
   <rij-variabelenaam> . <attribuutnaam> *

<vergelijkings-operator> ::= = | < | > | >= | <= | <>
```

De *conditie* in de calculus is bijna hetzelfde als in de algebra. Een conditie begint met een attribuutnaam gevolgd door een willekeurige vergelijkingsoperator (=, >, <, >=, <= of <>; zie paragraaf 2.3 voor een toelichting). Zij wordt afgesloten met een constante of een attribuutnaam. Vóór elke attribuutnaam *moet* de naam van de variabele genoemd worden. Het verschil met de relationele algebra is dat vóór de attribuutnaam niet een relatienaam gespecificeerd wordt maar de naam van een variabele. We geven een paar voorbeelden.

Voorbeeld 4.5: Geef alle werknemers die in Voorburg wonen.

```
(W.WNR, W.NAAM, W.ANR, W.SALARIS, W.WOONPLAATS,
 W.GEB-PLAATS WHERE W.WOONPLAATS = 'Voorburg')
```

Resultaat:

```
WNR  NAAM         ANR  SALARIS  WOONPLAATS  GEB-PLAATS
---  -----------  ---  -------  ----------  ----------
 25  Hondijk      108    40000  Voorburg    ?
 70  Osewoudt     108    42000  Voorburg    Rijswijk
```

Voorbeeld 4.6: Geef alle werknemers die in hun geboorteplaats wonen.

```
(W.WNR, W.NAAM, W.ANR, W.SALARIS, W.WOONPLAATS,
 W.GEB-PLAATS WHERE W.WOONPLAATS = W.GEB-PLAATS)
```

Resultaat:

```
WNR  NAAM         ANR  SALARIS  WOONPLAATS  GEB-PLAATS
---  -----------  ---  -------  ----------  ----------
 10  Dingelam     104    70000  Amsterdam   Amsterdam
```

Toelichting: Net als bij de relationele algebra geldt voor de relationele calculus dat NULL-waarden niet aan een conditie voldoen, zie paragraaf 2.3.

Voorbeeld 4.7: Geef het nummer en het salaris van elke werknemer die meer dan € 50.000,- verdient.

```
(W.WNR, W.SALARIS  WHERE W.SALARIS > 50000)
```

Resultaat:

```
WNR  SALARIS
---  -------
 10    70000
 15    60000
 45    75000
 80    62000
 90    55000
```

Voorbeeld 4.8: Geef de naam van elke werknemer gevolgd door de naam van de afdeling waarvoor hij of zij werkt.

```
(W.NAAM, A.NAAM WHERE W.ANR = A.ANR)
```

Resultaat:

```
NAAM         NAAM
-----------  --------
Dingelam     Inkoop
Homme        Verkoop
Hondijk      Magazijn
Alberegt     Verkoop
Osewoudt     Magazijn
Geyerstein   Inkoop
Hermans      Magazijn
```

De evaluatie van de expressie vindt plaats in drie stappen:

1. Het cartesisch product van alle variabelen die voor de formule staan wordt gecreëerd.
2. Alle rijen waarvoor de formule waar is worden uit het cartesisch product geselecteerd (een horizontale deelverzameling).
3. De gespecificeerde attributen worden geselecteerd (een verticale deelverzameling).

4.5 Samengestelde formules met meer condities

Een formule mag bestaan uit een simpele conditie of een samengestelde conditie. Een simpele conditie bestaat uit een attribuut dat vergeleken wordt met een waarde of een attribuut. Samengestelde condities bestaan uit simpele condities, gekoppeld met boolean-operatoren. In de relationele calculus geven we de boolean-operatoren met de woorden AND, OR en NOT weer.

Voorbeeld 4.9: Geef het nummer, het salaris en de woonplaats van elke werknemer die meer dan € 40.000,- verdient èn in Voorburg woont.

```
(W.WNR, W.SALARIS, W.WOONPLAATS
   WHERE W.SALARIS > 40000
   AND   W.WOONPLAATS = 'Voorburg')
```

Resultaat:

```
WNR  SALARIS  WOONPLAATS
---  -------  ----------
 70    42000  Voorburg
```

Voorbeeld 4.10: Geef het nummer, het salaris en de woonplaats van elke werknemer die meer dan € 40.000,- verdient *of* in Voorburg woont.

```
(W.WNR, W.SALARIS, W.WOONPLAATS
   WHERE W.SALARIS > 40000
   OR    W.WOONPLAATS = 'Voorburg')
```

Resultaat:

```
WNR  SALARIS  WOONPLAATS
---  -------  ----------
 10    70000  Amsterdam
 15    60000  Wateringen
 25    40000  Voorburg
 45    75000  Schiedam
 70    42000  Voorburg
 80    62000  Amstelveen
 90    55000  Rijswijk
```

Voorbeeld 4.11: Geef de naam van elke werknemer die meer dan € 50.000,- verdient, gevolgd door de naam van zijn of haar afdeling.

```
(W.NAAM, A.NAAM
   WHERE W.SALARIS > 50000
   AND   W.ANR = A.ANR)
```

Resultaat:

```
NAAM         NAAM
-----------  --------
Dingelam     Inkoop
Homme        Verkoop
Alberegt     Verkoop
Geyerstein   Inkoop
Hermans      Magazijn
```

Toelichting: Let op de conditie `W.ANR = A.ANR`. Hiermee wordt in feite het equivalent van de equi-join gecreëerd.

Voorbeeld 4.12: Geef van elk diploma dat na 1985 verstrekt is en dat hoort bij een cursus die gedoceerd is door de heer Palm, de naam van de cursus, de tijdsduur van de cursus, de naam van de werknemer en de datum.

```
(C.NAAM, C.DUUR, W.NAAM, D.DATUM
   WHERE D.DIPLOMA >= 860000
   AND   C.LERAAR = 'Palm'
   AND   D.WNR = W.WNR
   AND   D.CNR = C.CNR)
```

Resultaat:

```
NAAM         DUUR  NAAM        DATUM
-----------  ----  ----------  ------
Organiseren     4  Dingelam    861220
Statistiek     10  Hermans     860514
Presenteren     4  Homme       860630
```

Voorbeeld 4.13: Geef het nummer en de woonplaats van elke werknemer die *niet* in Voorburg woont.

```
(W.WNR, W.WOONPLAATS WHERE NOT (W.WOONPLAATS = 'Voorburg'))
```

Resultaat:

```
WNR  WOONPLAATS
---  ----------
 10  Amsterdam
 15  Wateringen
 45  Schiedam
 80  Amstelveen
 90  Rijswijk
```

Uiteraard kan deze expressie ook zonder een NOT-operator geformuleerd worden:

```
(W.WNR, W.WOONPLAATS WHERE W.WOONPLAATS <> 'Voorburg')
```

Voorbeeld 4.14: Geef de naam, de woonplaats en het salaris van elke werknemer die niet in Amsterdam woont en niet meer dan € 60.000,- verdient.

```
(W.NAAM, W.WOONPLAATS, W.SALARIS
   WHERE NOT (W.WOONPLAATS = 'Amsterdam'
         AND W.SALARIS > 60000))
```

Resultaat:

```
NAAM         SALARIS  WOONPLAATS
-----------  -------  ----------
Homme          60000  Wateringen
Hondijk        40000  Voorburg
Osewoudt       42000  Voorburg
Hermans        55000  Rijswijk
```

4.6 Formules met de existentiequantor

De formulevormen die in de vorige paragrafen behandeld zijn, zijn alle afgeleid uit de propositielogica. De relationele calculus staat toe dat quantoren uit de predikatenlogica gebruikt worden. We zullen er drie behandelen: de existentiequantor, de alquantor en de implicatie. We beginnen met de *existentiequantor*. We introduceren deze vorm stap voor stap en met de volgende vraag.

Voorbeeld 4.15: Geef de naam, het nummer en het afdelingsnummer van de werknemers die in de verkoopafdeling werken.

De expressie kan als volgt geformuleerd worden:

```
(W.NAAM, W.WNR, A.ANR
   WHERE W.ANR = A.ANR
   AND   A.NAAM = 'Verkoop')
```

Resultaat:

```
NAAM         WNR  ANR
-----------  ---  ---
Homme         15  102
Alberegt      45  102
```

Veronderstel nu dat het ANR-attribuut niet in het resultaat opgenomen moet worden. Dit betekent dat A.ANR weggelaten moet worden. Echter, hoe moet dan gecontroleerd worden dat een bepaalde werknemer in de verkoopafdeling werkt? Voor dergelijke vragen bestaat de existentiequantor. De bovenstaande expressie ziet er als volgt uit wanneer de existentiequantor gebruikt wordt:

```
(W.NAAM, W.WNR
   WHERE EXISTS A (waarin W werkt
                   AND met de naam Verkoop))
```

De formule `EXISTS A (...)` moet gelezen worden als: 'er moet minstens één afdeling bestaan die voldoet aan de formule die tussen de haakjes staat'. De twee formules 'waarin W werkt' en 'met de naam Verkoop' zijn respectievelijk met de condities `W.ANR = A.ANR` en `A.NAAM = 'Verkoop'` weer te geven. De expressie wordt dan:

```
(W.NAAM, W.WNR
   WHERE EXISTS A (A.ANR = W.ANR
                   AND A.NAAM = 'Verkoop'))
```

Resultaat:

```
NAAM         WNR
-----------  ---
Homme         15
Alberegt      45
```

Toelichting: Voor elke rij in de WERKNEMER-relatie wordt gekeken of er minstens één rij in de AFDELING-relatie bestaat met hetzelfde afdelingsnummer als de betreffende werknemer (`A.ANR = W.ANR`) en met de naam 'Verkoop' (`A.NAAM = 'Verkoop'`).

Laten we de rijen van de WERKNEMER-relatie eens één voor één bekijken. Eerst wordt bepaald of de formule waar is voor de rij die betrekking heeft op werknemer Dingelam met afdelingsnummer 104. De formule is waar indien er in de AFDELING-relatie een rij bestaat met een afdelingsnummer gelijk aan 104 en met de naam Verkoop. Dit is onwaar. Conclusie: werknemer Dingelam werkt niet in de verkoopafdeling en komt dus niet in het resultaat voor. Vervolgens wordt de tweede rij met gegevens over Homme, werkend voor afdeling 102, geëvalueerd. Er bestaat inderdaad een rij in de AFDELING-relatie waar de waarde van het ANR-attribuut gelijk is aan dat van Homme en waarvan de naam gelijk is aan 'Verkoop'. Homme werkt dus wel voor de verkoopafdeling en wordt in het resultaat opgenomen. Op deze wijze wordt elke rij geëvalueerd met bovenstaande relatie als resultaat.

Voorbeeld 4.16: Geef de namen van de afdelingen die een werknemer in dienst hebben die in Amsterdam geboren is.

```
(A.NAAM WHERE
   EXISTS W (W.ANR = A.ANR
             AND W.GEB-PLAATS = 'Amsterdam'))
```

Resultaat:

```
NAAM
-----------
Inkoop
Magazijn
```

Toelichting: Geef de naam van elke afdeling waarvoor minstens één werknemer bestaat (`EXISTS W`) die in die afdeling werkt (`W.ANR = A.ANR`) 'en die in Amsterdam geboren is (`W.GEB-PLAATS = 'Rotterdam'`).

Voorbeeld 4.17: Geef de namen van de afdelingen die werknemer 90 als baas hebben en minstens één werknemer in dienst hebben die in Amsterdam geboren is.

```
(A.NAAM WHERE A.BAAS = 90 AND
   EXISTS W (W.ANR = A.ANR
             AND W.GEB-PLAATS = 'Amsterdam'))
```

Resultaat:

```
NAAM
-----------
Magazijn
```

Voorbeeld 4.18: Geef de namen van de werknemers die cursus 30 gevolgd hebben en tot een afdeling behoren die in Rotterdam gevestigd is.

```
(W.NAAM WHERE
   EXISTS D (D.WNR = W.WNR AND D.CNR = 30)
      AND
   EXISTS A (A.ANR = W.ANR AND A.LOKATIE = 'Rotterdam'))
```

Resultaat:

```
NAAM
-----------
Homme
Alberegt
```

Voorbeeld 4.19: Geef de namen van de werknemers die een cursus gevolgd hebben die door Palm gedoceerd wordt.

```
(W.NAAM WHERE
   EXISTS D (D.WNR = W.WNR AND
      EXISTS C (C.CNR = D.CNR AND C.LERAAR = 'Palm')))
```

Resultaat:

```
NAAM
-----------
Homme
Alberegt
```

Laten we deze expressie eens stap voor stap opbouwen. De vraag kan ook anders gesteld worden: 'Geef de naam van werknemer W als er een diploma D bestaat dat uitgereikt is aan W en er een cursus bestaat voor diploma D gedoceerd door Palm'. De expressie is nu te formuleren met behulp van een combinatie van relationele calculus en nederlandstalige zinnen:

```
(W.NAAM WHERE
   EXISTS D (uitgereikt aan W AND
             behorend bij een cursus gedoceerd door Palm)
```

De conditie 'uitgereikt aan W' kan vervangen worden door `D.WNR = W.WNR`. De conditie 'behorend bij een cursus gedoceerd door Palm' is zelf weer te vervangen door twee andere zinnen:

```
(W.NAAM WHERE
   EXISTS D (D.WNR = W.WNR AND
      EXISTS C (behorend bij het diploma AND
                gedoceerd door Palm)
```

De twee overgebleven zinnen zijn nu eenvoudig te vervangen door formele condities. De uiteindelijke expressie is hierboven reeds weergegeven.

Een formule met de volgende vorm (R1 en R2 zijn twee variabelen):

```
EXISTS R1 ( conditie-1 AND EXISTS R2 ( conditie-2 ))
```

is per definitie gelijkwaardig aan:

```
EXISTS R1 ( EXISTS R2 ( conditie-1 AND conditie-2 ))
```

Een andere gelijkwaardige formulering is:

```
EXISTS R2 ( EXISTS R1 ( conditie-1 AND conditie-2 ))
```

De eerste expressie in voorbeeld 4.19 kan daarom ook als volgt geformuleerd worden:

```
(W.NAAM WHERE
   EXISTS D ( EXISTS C (D.WNR = W.WNR AND
                        C.CNR = D.CNR AND
                        C.LERAAR = 'Palm')))
```

Voorbeeld 4.20: Geef de namen van de werknemers die *niet* op een Rotterdamse afdeling werken.

```
(W.NAAM WHERE
   NOT EXISTS A (W.ANR = A.ANR AND
                 A.LOKATIE = 'Rotterdam'))
```

Resultaat:

```
NAAM
-----------
Dingelam
Hondijk
Osewoudt
Geyerstein
Hermans
```

Toelichting: Geef de namen van de werknemers voor wie geldt dat er *geen* afdeling bestaat met een afdelingsnummer gelijk aan dat van de betreffende werknemer en die in Rotterdam gevestigd is.

De expressie kan ook geformuleerd worden zonder gebruik te maken van NOT EXISTS:

```
(W.NAAM WHERE
   EXISTS A (W.ANR = A.ANR AND A.LOKATIE <> 'Rotterdam'))
```

Voorbeeld 4.21: Geef de namen van de werknemers die *alle* cursussen gevolgd hebben.

```
(W.NAAM WHERE
   NOT EXISTS C (NOT EXISTS D (C.CNR = D.CNR AND W.WNR = D.WNR)))
```

Resultaat:

```
NAAM
-----------
Homme
```

Toelichting: Deze expressie is als volgt weer te geven: 'Geef de naam van elke werknemer waarvoor *geen* cursus bestaat die hij/zij *niet* gevolgd heeft.' Deze instructie kan eenvoudiger geformuleerd worden, maar daar komen we in de volgende paragraaf op terug.

Voorbeeld 4.22: Geef de naam en het salaris van elke werknemer die meer verdient dan Hermans.

Om de vraag te kunnen beantwoorden is de W2-variabele nodig (zie begin paragraaf 4.3):

```
(W.NAAM, W.SALARIS WHERE
   EXISTS W2 (W2.NAAM = 'Hermans' AND W2.SALARIS < W.SALARIS))
```

Resultaat:

```
NAAM         SALARIS
-----------  -------
Dingelam       70000
Homme          60000
Alberegt       75000
Geyerstein     62000
```

Toelichting: Geef de naam en het salaris van elke werknemer, waarvoor een andere werknemer bestaat (`EXISTS W2`) genaamd Hermans (`W2.NAAM = 'Hermans'`) en die meer dan hem verdient (`W2.SALARIS < W.SALARIS`).

Voorbeeld 4.23: Geef het nummer en de naam van elke werknemer die minstens alle cursussen gevolgd heeft die ook door werknemer 45 gevolgd zijn.

```
(W.WNR, W.NAAM WHERE
   NOT EXISTS D (D.WNR = 45 AND
                 NOT EXISTS D2 (D2.WNR = W.WNR AND D.CNR = D2.CNR)))
```

Resultaat:

```
WNR  NAAM
---  --------
 15  Homme
 45  Alberegt
```

Toelichting: Geef het nummer en de naam van elke werknemer voor wie geen diploma bestaat (`NOT EXISTS D`) dat door werknemer 45 behaald is (`D.WNR = 45`) en waarvoor geen ander diploma bestaat (`NOT EXISTS D2`) dat aan dezelfde werknemer uitgereikt is (`D2.WNR = W.WNR`) en betrekking heeft op dezelfde cursus (`D.CNR = D2.CNR`).

Voorbeeld 4.24: Geef het nummer en de naam van elke werknemer die minstens één cursus heeft gevolgd die ook door werknemer 45 gevolgd is.

```
(W.WNR, W.NAAM WHERE
   EXISTS D (D.WNR = W.WNR AND
      EXISTS D2 (D.CNR = D2.CNR AND D2.WNR = 45)))
```

Resultaat:

```
WNR  NAAM
---  -----------
 10  Dingelam
 15  Homme
 45  Alberegt
```

Als werknemer 45 zelf niet in het resultaat behoort voor te komen, dan moet de expressie als volgt uitgebreid worden:

```
(W.WNR, W.NAAM WHERE W.WNR <> 45 AND
   NOT EXISTS D (D.WNR = W.WNR AND
      EXISTS D2 (D.CNR = D2.CNR AND D2.WNR = 45)))
```

Voorbeeld 4.25: Geef de naam van elke werknemer die in de verkoop- of inkoopafdeling werkt; geef ook de naam van de afdeling.

```
(W.NAAM, A.NAAM WHERE W.ANR = A.ANR AND
   EXISTS A2 (A2.ANR = W.ANR AND (A2.NAAM = 'Verkoop' OR A2.NAAM = 'Inkoop')))
```

Resultaat:

```
NAAM         NAAM
-----------  --------
Dingelam     Inkoop
Homme        Verkoop
Alberegt     Verkoop
Geyerstein   Inkoop
```

Een alternatieve formulering is:

```
(W.NAAM, A.NAAM WHERE W.ANR = A.ANR AND
   (A2.NAAM = 'Verkoop' OR A2.NAAM = 'Inkoop'))
```

4.7 Formules met de alquantor

In principe kan elke vraag omgezet worden in een calculus-expressie, waarbij alleen gebruik gemaakt wordt van wat in de vorige paragrafen behandeld is. Maar net zoals de predikatenlogica uitgebreid is met de alquantor en de implicatie, is de relationele calculus dit ook. De alquantor en de implicatie vergroten de expressiemogelijkheden van de relationele calculus dus niet.

Voorbeeld 4.26: Geef de naam en het salaris van de werknemer die het meest verdient. Als er toevallig meerdere werknemers het 'hoogste' salaris hebben, worden ze allemaal in het resultaat opgenomen.

```
(W.NAAM, W.SALARIS WHERE
   NOT EXISTS W2 (W2.SALARIS > W.SALARIS))
```

Resultaat:

```
NAAM         SALARIS
-----------  -------
Alberegt       75000
```

De bovenstaande expressie komt 'natuurlijker' over als de *alquantor* gebruikt wordt:

```
(W.NAAM, W.SALARIS WHERE
   FORALL W2 (W2.SALARIS <= W.SALARIS))
```

De expressie moet als volgt gelezen worden: 'Geef de naam en het salaris van werknemer W, als voor alle werknemers geldt dat hun salaris kleiner of gelijk is aan dat van W'.

De definitie:

```
FORALL r (conditie)
```

is gelijkwaardig aan:

```
NOT EXISTS r (NOT conditie)
```

Voorbeeld 4.27: Geef de namen van de werknemers die *alle* cursussen gevolgd hebben (zie voorbeeld 4.21).

```
(W.NAAM WHERE
   FORALL C (EXISTS D (C.CNR = D.CNR AND W.WNR = D.WNR)))
```

Resultaat:

```
NAAM
-----------
Homme
```

Toelichting: 'Geef de naam van elke werknemer W, als voor elke cursus C geldt dat er een diploma D bestaat dat betrekking heeft op cursus C en dat uitgereikt is aan W'.

Voorbeeld 4.28: Geef het nummer en de naam van elke werknemer die tenminste *alle* cursussen gevolgd heeft die ook door werknemer 45 gevolgd zijn (zie voorbeeld 4.23).

```
(W.WNR, W.NAAM WHERE
   FORALL D (D.WNR <> 45 OR EXISTS D2
      (D2.CNR = D.CNR AND W.WNR = D2.WNR)))
```

Resultaat:

```
WNR  NAAM
---  -----------
 15  Homme
```

Toelichting: 'Geef het nummer en de naam van elke werknemer W als voor elk diploma D geldt: of het is géén diploma van werknemer 45, of, als het wel een diploma van werknemer 45 is, er bestaat ook een ander diploma dat uitgereikt is aan W en betrekking heeft op dezelfde cursus als diploma D'.

In een paar stappen laten we zien hoe deze expressie met een alquantor omgezet kan worden naar de expressie in voorbeeld 4.23 waarin existentiequantoren gebruikt worden. Voor het gemak vervangen we de formule `EXISTS D2 (D2.CNR = D.CNR AND W.WNR = D2.WNR)` door f1.

```
(W.WNR, W.NAAM WHERE
   FORALL D (D.WNR <> 45 OR f1))
```

Stap 1:

```
(W.WNR, W.NAAM WHERE
   NOT EXISTS D (NOT(D.WNR <> 45 OR f1)))
```

Stap 2:

```
(W.WNR, W.NAAM WHERE
   NOT EXISTS D (NOT(D.WNR <> 45) AND NOT f1)))
```

Stap 3:

```
(W.WNR, W.NAAM WHERE
   NOT EXISTS D (D.WNR = 45 AND NOT f1))
```

Tenslotte vervangen we f1 weer door de oorspronkelijke formule; de expressie bevat nu alleen nog maar existentiequantoren:

```
(W.WNR, W.NAAM WHERE
   NOT EXISTS D (D.WNR = 45 AND
      NOT EXISTS D2 (D2.CNR = D.CNR AND W.WNR = D2.WNR)))
```

4.8 Formules met de implicatie

Net als de alquantor vergroot de *implicatie* de expressiemogelijkheden van de relationele calculus niet. Gebruik ervan vereenvoudigt alleen het formuleren van bepaalde expressies. We geven direct een voorbeeld.

Voorbeeld 4.29: Geef het nummer, de naam, de afdeling en het salaris van elke werknemer, behalve als hij of zij in afdeling 108 werkt en minder dan € 50.000,- verdient.

```
(W.WNR, W.NAAM, W.ANR, W.SALARIS WHERE
   NOT (W.ANR = 108) OR W.SALARIS > 50000)
```

Resultaat:

```
WNR  NAAM         ANR  SALARIS
---  -----------  ---  -------
 10  Dingelam     104    70000
 15  Homme        102    60000
 45  Alberegt     102    75000
 80  Geyerstein   104    62000
 90  Hermans      108    55000
```

De formulering van de expressie lijkt niet op de oorspronkelijke vraag. De expressie lijkt enigszins 'onnatuurlijk'. Daarom maken we gebruik van de implicatie. Een formule met de volgende vorm (f1 en f2 zijn allebei formules):

```
(NOT f1) OR f2
```

is gelijkwaardig aan:

```
IF f1 THEN f2
```

De bovenstaande implicatie betekent dat als f1 waar is, f2 ook waar moet zijn. Met andere woorden, de bovenstaande formule is alleen waar als f1 en f2 beide waar zijn of als f1 onwaar is. De *IF THEN-constructie* heeft hier dus een andere betekenis dan een *IF THEN-instructie* in programmeertalen als Pascal en COBOL. In dit soort talen wordt achter THEN een actie gespecificeerd. In de relationele calculus staat achter de IF en achter de THEN een formule.

Nu we bekend zijn met de implicatie, kunnen we een andere formulering voor de expressie van voorbeeld 4.29 geven die meer op de oorspronkelijke vraag lijkt:

```
(W.WNR, W.NAAM, W.ANR, W.SALARIS WHERE
   IF W.ANR = 108 THEN W.SALARIS > 50000)
```

Voorbeeld 4.30: Geef de nummers en de namen van de cursussen die alleen door werknemer 15 gevolgd zijn.

Als we de implicatie niet gebruiken, krijgen we de volgende expressie:

```
(C.CNR, C.NAAM WHERE
   FORALL D (D.CNR <> C.CNR OR D.WNR = 15))
```

Resultaat:

```
CNR  NAAM
---  -----------
 20  Presenteren
```

Met gebruik van de implicatie:

```
(C.CNR, C.NAAM WHERE
   FORALL D (IF D.CNR = C.CNR THEN D.WNR = 15))
```

Toelichting: 'Geef het nummer en de naam van elke cursus C, als voor elk diploma D geldt dat als het betrekking heeft op cursus C, het dan uitgereikt moet zijn aan werknemer 15'.

Voorbeeld 4.31: Geef het nummer en de naam van elke werknemer die tenminste *alle* cursussen gevolgd heeft die ook door werknemer 45 gevolgd zijn (zie voorbeelden 4.23 en 4.28).

```
(W.WNR, W.NAAM WHERE
   FORALL D (IF D.WNR = 45 THEN
      EXISTS D2 (D2.CNR = D.CNR AND D2.WNR = W.WNR)))
```

Resultaat:

```
WNR  NAAM
---  -----------
 15  Homme
```

Voorbeeld 4.32: Geef de namen van de werknemers die alle cursussen gevolgd hebben die door leraar Palm gedoceerd worden.

```
(W.NAAM WHERE
   FORALL C (IF C.LERAAR = 'Palm'
             THEN EXISTS D (D.CNR = C.CNR
                            AND D.WNR = W.WNR)))
```

Resultaat:

```
NAAM
-----------
Homme
Alberegt
```

Toelichting: 'Geef de naam van elke werknemer W als voor elke cursus C geldt dat als de docent Palm heet, er dan een diploma D voor de cursus C moet bestaan die uitgereikt is aan werknemer W'.

4.9 De rij-variabele (vervolg)

In paragraaf 4.2 zijn voorbeelden gegeven van simpele definities van rij-variabelen. Deze paragraaf beschrijft hoe een rij-variabele met een calculus-expressie gedefinieerd kan worden.

Voorbeeld 4.33: Geef alle werknemers die in Voorburg wonen.

In vraag 7.3.1 is voor deze vraag de volgende expressie geformuleerd:

```
(W.WNR, W.NAAM, W.ANR, W.SALARIS, W.WOONPLAATS,
 W.GEB-PLAATS WHERE W.WOONPLAATS = 'Voorburg')
```

We kunnen de vraag echter ook op de volgende (enigszins omslachtige) wijze beantwoorden.

```
VAR VBG : W.WNR, W.NAAM, W.ANR, W.SALARIS, W.WOONPLAATS,
          W.GEB-PLAATS WHERE W.WOONPLAATS = 'Voorburg'

( VBG )
```

Resultaat:

```
WNR  NAAM         ANR  SALARIS  WOONPLAATS  GEB-PLAATS
---  -----------  ---  -------  ----------  ----------
 25  Hondijk      108    40000  Voorburg    ?
 70  Osewoudt     108    42000  Voorburg    Rijswijk
```

De definitie van de VBG-variabele bestaat niet uit de naam van een relatie, zoals in alle vorige voorbeelden, maar uit een volledige calculus-expressie. Het resultaat van deze expressie vormt nu het domein van de VBG-variabele.

Voorbeeld 4.34: Geef de namen van de werknemers die in Voorburg wonen.

```
(VBG.NAAM)
```

Resultaat:

```
NAAM
----------
Hondijk
Osewoudt
```

Voorbeeld 4.35: Geef de verschillende plaatsnamen die in het WOONPLAATS-attribuut van de WERKNEMER-relatie en in het LOKATIE-attribuut van de AFDELING-relatie staan (zie voorbeeld 2.11).

```
VAR PLAATS : (W.WOONPLAATS), (W.LOKATIE)

(PLAATS)
```

Resultaat:

```
WOONPLAATS
----------
Amsterdam
Wateringen
Voorburg
Schiedam
Amstelveen
Rijswijk
Rotterdam
```

Voorbeeld 4.36: Geef de namen en de woonplaatsen van de werknemers die in Wateringen en Voorburg wonen (zie voorbeeld 2.13).

```
VAR WV : (W.WNR, W.NAAM, ..., W.GEB-PLAATS
          WHERE W.WOONPLAATS = 'Wateringen'),
         (W.WNR, W.NAAM, ..., W.GEB-PLAATS
          WHERE W.WOONPLAATS = 'Voorburg')

(WV.NAAM, WV.WOONPLAATS)
```

Resultaat:

```
NAAM       WOONPLAATS
---------- ----------
Homme      Wateringen
Hondijk    Voorburg
Osewoudt   Voorburg
```

4.10 Semantic override

Het begrip *semantic override* is reeds in paragraaf 2.9 besproken. Daar werd besproken hoe semantic override in algebraïsche expressies te forceren is. Het forceren van semantic override in calculus-expressies gebeurt op identieke wijze.

Voorbeeld 4.37: Geef elke werknemer waarvan de naam gelijk is aan de naam van zijn of haar woonplaats, zie vraag 5.8.1.

```
(W WHERE W.NAAM* = W.WOONPLAATS*)
```

En uiteraard is ook nu het resultaat leeg.

Voorbeeld 4.38: Geef een lijst met namen van werknemers en leraren, zie voorbeeld 2.17.

Hiervoor moet een aparte variabele gedefinieerd worden:

```
VAR N : W.NAAM*, C.LERAAR*
```

De expressie wordt nu:

```
(N)
```

Resultaat:

```
NAAM
----------
Dingelam
Homme
Hondijk
Alberegt
Osewoudt
Geyerstein
Hermans
Palm
Prater
Cools
Veen
```

4.11 Vrije en gebonden variabelen

In de vorige paragrafen zijn we voorbijgegaan aan het zogenaamde *bereik* van een rij-variabele. Variabelen die vóór de formule gespecificeerd zijn, mogen overal in de calculus-expressie gebruikt worden. In de volgende expressie mag de V-variabele bijvoorbeeld overal gebruikt worden, ook in de formule f1.

```
(V.A WHERE .... AND EXISTS W ( f1 ))
```

V wordt in dit geval een *vrije variabele* (free variable) genoemd. De W-variabele mag echter alleen in de f1-formule gebruikt worden. Het bereik van deze variabele is dus beperkt tot f1 en wordt daarom een *gebonden*

variabele (bound variable) genoemd. De volgende expressie is incorrect, omdat W alleen buiten de f2-formule gebruikt mag worden.

```
(V.A WHERE W.A = .... AND EXISTS W ( f2 ))
```

Het bereik van de V-variabele is groter dan die van de W-variabele. Het principe van vrije en gebonden variabelen is te vergelijken met respectievelijk globale en lokale variabelen in programmeertalen als ADA en Pascal.

De Auteur

Rick F. van der Lans is auteur van veel boeken over SQL. Naast dit SQL Leerboek dat in diverse talen vertaald is, waaronder Engels, Duits, Chinees enItaliaans, heeft hij SQL boeken geschreven voor producten als MySQL, Oracle, SQLite, Ingres en Pervasive PSQL.

Hij is onafhankelijk adviesur, auteur en docent gespecialiseerd in databasetechnologie, datawarehousing en applicatie-integratie. Hij is oprichter en directeur van R20/Consultancy. Door de jaren heen heeft hij veel organisaties geadviseerd op het gebied van IT-architecturen.

Als spreker op conferenties en seminars wordt hij internationaal gerespecteerd. Al meer dan vijfentwintig jaar geeft hij over de gehele wereld lezingen, inclusief in de meeste Europese landen, Noord- en Zuid-Amerika en Australië. Hij is voorzitter van het jaarlijkse European Data Warehouse and Business Intelligence Conference. Hij schrijft een column voor Database Magazine en voor het internationale Beye-Network.com. Zeven jaar lang was hij lid van de Nederlandse ISO commissie verantwoordelijk voor ISO SQL Standaard.

Rick kan via de volgende kanalen bereikt worden:

Email: rick@r20.nl
Twitter: http://twitter.com/Rick_vanderlans
LinkedIn: http://www.linkedin.com/pub/rick-van-der-lans/9/207/223

Cursussen over de volgende onderwerpen kunnen door Rick F. van der Lans verzorgd worden

- Database-ontwerp en informatiemodellering
- De basis van SQL
- Het ontwikkelen van geavanceerde SQL queries
- Datawarehousing en business intelligence
- Data virtualisatie

Andere boeken geschreven door Rick F. van der Lans