LIMBURGS UNIVERSITAIR CENTRUM
FACULTEIT TOEGEPASTE ECOMOMISCHE WETENSCHAPPEN

Gevalstudie: Toepassing van een markeertaal op een data-integratiesysteem

Thesis voorgedragen tot
het behalen van de graad van
Handelsingenieur in de Beleidsinformatica.

Door: Lene Clijsters

Promotor: Prof. Dr. Koen Vanhoof

Academiejaar: 2004-2005

Voorwoord

Toen de mogelijkheid bestond om een thesis te schrijven rond de toepassing van een markeertaal op een data-integratiesysteem, liet ik deze uitzonderlijke kans niet aan mijn neus voorbijgaan. Het resultaat van mijn werk is volgend proefschrift.

Dit proefschrift zou nooit als dusdanig tot stand zijn gekomen zonder de hulp van Ronny Martens en Yves Sterckx van het bedrijf IKAN Software. Zowel de toepassing ETL4ALL als de supplementaire kennis over datawarehousing werden aangreikt door deze mensen. Een speciale vermelding gaat uit naar mijn vader, Iso Clijsters, die geholpen heeft bij het zoeken naar data voor het toepassen van de gevalstudie. Zo ook Cindy Ramant en Lode Maris van Sage BOB Software, die een bijdrage hebben geleverd bij het uitvoeren van de analyse. Mijn promotor Prof. Dr. Koen Vanhoof mag zeker niet worden vergeten. Door het voorzien van de juiste kennis op het juiste moment heeft hij het groeiproces van deze thesis steeds in goede banen weten te leiden. Als laatste zou ik mijn ouders, mijn broer en mijn vriend willen bedanken omdat ik steeds op hun volle steun kan rekenen.

Samenvatting

De afgelopen jaren hebben ondernemingen in sterk toenemende mate geïnvesteerd in ERP(Enterprise Resource Planning)-systemen. ERP integreert alle processen en functies van een organisatie met als doel het presenteren van een holistisch beeld vanuit één enkele informatiearchitectuur. De beschikbaarheid van bedrijfsinformatie op operationeel niveau wordt op die manier aanzienlijk vergroot. Van daaruit ontstaat vaak de verwachting dat de invoering van een ERP-pakket ook belangrijke voordelen kan hebben voor de voorziening van managementinformatie. Maar hoewel de verzamelde gegevens potentiële voordelen bieden voor het verbeteren van managementinformatie, wordt geen verbetering in de managementrapportage bewerkstelligd door het verzamelen van deze gegevens in een ERP-systeem.

Indien ondernemingen daadwerkelijk voordeel willen halen uit hun ERP-pakket voor de voorziening van managementinformatie, dienen ze het ontsluiten van de ERP-gegevensbron prioriteit te geven. Hiervoor kan men gebruik maken van een Business Intelligence (BI) toepassing. BI heeft als doelstelling het transformeren van gegevens naar informatie en deze informatie vervolgens om te zetten in kennis. In dit domein situeert zich Datamining. Datamining is een methode voor het zoeken naar interessante informatie in grote hoeveelheden gegevens. Met behulp van deze informatie kunnen trends en verbanden worden ontdekt die bedrijven de mogelijkheid bieden om proactief kennisgedreven beslissingen te nemen.

Deze eindverhandeling onderzoekt de uitvoerbaarheid van het integreren van een datamining model in een ERP-systeem aan de hand van een concrete gevalstudie. Specifiek wordt gebruik gemaakt van de recent ontwikkelde PMML standaard. PMML of Predictive Model Markup Language voorziet een methode om datamining modellen te definiëren en uit te wisselen tussen diverse toepassingen. Enerzijds wordt nagegaan op welke wijze een datamining model kan worden geïntegreerd in een ERP-systeem door middel van PMML. Anderzijds wordt onderzocht in welke mate het integreren van een PMML model mogelijk is in de praktijk met behulp van hedendaagse toepassingen.

In de uitwerking van de gevalstudie wordt eerst een grondige analyse gemaakt van de databasestructuur van een ERP-systeem. Hieruit zal blijken dat datamining niet rechtstreeks kan

worden toegepast op de gegevens in dit systeem. Het is namelijk ontworpen voor het vastleggen van transacties en biedt daarom geen stabiele basis voor datamining. Er wordt een oplossing gevonden door het ontwikkelen van een datawarehouse die naast de bestaande ERP-database functioneert en afgestemd is op de behoeften van een analysetoepassing. Een datawarehouse is een platform met geïntegreerde, kwalitatief hoogwaardige gegevens. Het ontwerp van het datawarehouse wordt gebaseerd op een dimensionaal gegevensmodel. Met behulp van een ETL(Extract Transform Load)-toepassing kunnen de benodigde gegevens uit de ERP-database worden geëxtraheerd en vervolgens getransformeerd naar de gewenste vorm. Deze gegevens worden nadien in het datawarehouse geladen.

Op basis van de gegevens in het datawarehouse wordt vervolgens een tabel samengesteld bestaande uit observaties en variabelen. Deze tabel dient als gegevensbron voor de ontwikkeling van een datamining model met behulp van een datamining toepassing. Het resulterende model wordt tenslotte vertaald naar PMML code. De volgende stap is dan het importeren van dit model in het datawarehouse om het uiteindelijk toe te passen op de gegevens uit het ERP-systeem. Hierbij zal een belangrijke tekortkoming worden vastgesteld die een eigenlijke integratie verhindert. Dit resultaat kan echter niet worden veralgemeend daar het afhangt van de keuze van de datamining toepassing.

Algemeen wordt voor beide onderzoeksvragen een positief antwoord gevonden. Gebruik makend van een datawarehouse wordt de invloed van de beperkingen van een ERP-systeem geminimaliseerd. Hierdoor kan datamining alsnog worden toegepast op de gegevens van het ERP-systeem. Na het omzetten van het resulterende model in PMML code, kan het worden geïmplementeerd in het datawarehouse. Uiteindelijk zal het wel of niet slagen van de integratie afhangen van de PMML versie en de mate van ondersteuning door de datamining applicatie.

Inhoudsopgave

1 Probleemstelling **1**
1.1 Verkenning en verheldering van het praktijkprobleem . . . 1
1.1.1 Enterprise Resource Planning (ERP) . . . 1
1.1.2 Beperkingen van een ERP-systeem . . . 2
1.1.3 Business Intelligence . . . 2
1.2 Uitwerking van het onderzoeksprobleem . . . 3
1.2.1 Afleiding van de centrale onderzoeksvraag . . . 4
1.2.2 Strategie van het verkennend onderzoek . . . 4
1.3 Motivatie van het gekozen onderwerp . . . 5

I Literatuurstudie **6**

2 Enterprise Resource Planning **7**
2.1 Wat is Enterprise Resource Planning? . . . 7
2.1.1 Definitie . . . 7
2.1.2 Evolutie van ERP . . . 8
2.2 ERP-systemen in de organisatie . . . 9
2.3 Commerciële ERP-pakketten . . . 10
2.4 Toekomst van ERP . . . 11

3 Datawarehousing **12**
3.1 Wat is datawarehousing? . . . 12
3.1.1 Noodzaak van datawarehousing . . . 12
3.1.2 Definitie . . . 13
3.1.3 Doelstellingen . . . 14
3.2 Datawarehouse-architectuur . . . 15
3.2.1 Basiselementen . . . 15
3.2.1.1 Brongegevens . . . 16
3.2.1.2 Extraheren - Transformeren - Laden (ETL) . . . 16
3.2.1.3 Gegevensopslag . . . 16
3.2.1.4 Analyse en rapportage . . . 17
3.2.1.5 Metagegevens . . . 17
3.2.2 Top-down versus bottom-up benadering . . . 18
3.3 Dimensionaal gegevensmodel . . . 20
3.3.1 Algemeen . . . 20
3.3.2 Het doel van dimensionaal modelleren . . . 20

3.3.3 Het sterschema . . . 21
3.3.4 Voordelen van dimensionaal modelleren . . . 22
3.3.5 Het vier-stappen-model voor het ontwerpen van een dimensionaal model 22

4 Datamining **25**
4.1 Wat is datamining? . . . 25
4.2 De evolutie van datamining . . . 26
4.3 De voordelen van datamining . . . 28
4.4 Het datamining proces . . . 28
4.4.1 Algemeen . . . 28
4.4.2 Problemen in het datamining proces . . . 31
4.5 Datamining en datawarehousing . . . 31
4.6 Beslissingsbomen . . . 32

5 Predictive Model Markup Language **34**
5.1 eXtensible Markup Language (XML) . . . 34
5.1.1 Wat is XML? . . . 34
5.1.2 De structuur van een XML document . . . 36
5.1.3 Juist gevormde XML documenten . . . 38
5.1.4 XML validatie . . . 39
5.1.4.1 Validatie met behulp van Document Type Definition (DTD) . . . 39
5.1.4.2 Validatie op basis van een XML schema . . . 40
5.2 Predictive Model Markup Language (PMML) . . . 42
5.2.1 Wat is PMML? . . . 42
5.2.2 Algemene structuur van een PMML document . . . 42
5.2.3 Componenten van een PMML document . . . 43
5.2.3.1 Hoofding (Header) . . . 43
5.2.3.2 Gegevenswoordenboek (Data Dictionary) . . . 44
5.2.3.3 Transformatiewoordenboek (Transformation Dictionary) . . . 45
5.2.3.4 Mining Schema . . . 46
5.2.3.5 Locale transformaties (Local Transformations) . . . 47
5.2.3.6 Model parameters . . . 47

II Gevalstudie **50**

6 Analyse van de gegevensstructuur in Lijn BOB **51**
6.1 Lijn BOB . . . 51
6.2 Omschrijving van de informatiebehoefte . . . 52
6.3 Analyse van de databasestructuur . . . 52
6.4 Analyse van de gegevens . . . 54
6.4.1 De tabel IART . . . 54
6.4.2 De tabel COMPAN . . . 54
6.4.3 De tabel IHISTO . . . 55
6.4.4 De relaties tussen IART, COMPAN en IHISTO . . . 57
6.5 Beoordeling van de databasestructuur . . . 58

7 Constructie van een datamart met behulp van ETL4ALL **60**
7.1 Ontwerp van het dimensionaal gegevensmodel . . . 60
7.1.1 Stap 1: Het onderwerp selecteren van de datamart . . . 60
7.1.2 Stap 2: Het aangeven van de betekenis van de feitentabel . . . 61
7.1.3 Stap 3: De dimensies identificeren en conformeren . . . 61
7.1.4 Stap 4: De feiten kiezen . . . 61
7.1.5 De ster . . . 62
7.2 ETL4ALL . . . 62
7.2.1 Het doel van ETL4ALL . . . 62
7.2.2 De architectur van ETL4ALL . . . 63
7.3 Opvulling van de datamart met behulp van ETL4ALL . . . 64
7.3.1 Inleiding . . . 64
7.3.1.1 Het analyseren van de behoeften . . . 65
7.3.1.2 Het definiëren van een transformatieprogramma . . . 66
7.3.1.3 Het uitvoeren van een transformatieprogramma . . . 68
7.3.2 Stap 1: het importeren van de gegevens in ETL4ALL . . . 69
7.3.3 Stap 2: Het definiëren van procedures . . . 70
7.3.4 Stap 3: Het samenstellen van een programma . . . 72
7.3.5 Stap 4: Het uitvoeren van het programma via een virtuele gegevensbron 73
7.3.6 Stap 5: De resultaten naar een databasetabel sturen . . . 74
7.4 Samenstelling van een datamining tabel op basis van de datamart . . . 75
7.4.1 Inleiding . . . 75
7.4.2 Observaties en variabelen voor datamining . . . 76
7.4.3 Transformaties voor het berekenen van de variabelen . . . 76
7.4.3.1 Tussenstap 1: berekening van TrouwKlant en LaatsteAankoop . 77
7.4.3.2 Tussenstap 2: berekening van JaarProd en JaarWinst . . . 78
7.4.3.3 Berekening van de klantgegevens . . . 78
7.5 Conclusie na het gebruik van een ETL-toepassing . . . 79

8 Ontwikkeling van een datamining model in PMML met behulp van SAS Enterprise Miner **82**
8.1 Inleiding . . . 82
8.2 SEMMA . . . 83
8.2.1 Sample: Identificatie van de input datasets . . . 83
8.2.2 Explore: Verkennende gegevensanalyse . . . 84
8.2.3 Modify: Transformatie van de variabelen . . . 85
8.2.4 Model: Ontwikkeling van modellen . . . 87
8.2.5 Assess: Evaluatie van de modellen . . . 88
8.3 De beslissingsboom van de variabele JaarWinst . . . 89
8.4 Omzetting van het model naar PMML . . . 90

9 Conclusies **94**
9.1 Beantwoording van de centrale onderzoeksvraag . . . 94
9.1.1 Op welke wijze kan een datamining model geïmplementeerd worden in een ERP-systeem door toepassing van PMML? . . . 95
9.1.2 Is het integreren van een datamining model in een ERP-systeem door middel van PMML uitvoerbaar in de praktijk? . . . 95

9.2 Algemene conclusie 96

III Bijlagen **98**

A De tabel IART **99**

B De tabel COMPAN **102**

C De tabel IHISTO **104**

D SQL-declaraties van de datamart **106**

E Visuele voorstellingen van de transformaties in ETL4ALL voor het berekenen van de variabelen **108**

F Grafische analyse van de variabelen **113**

G Uitgebreide voorstelling van beslissingsboom vijf **115**

H Het PMML document van beslissingsboom vijf **118**

Lijst van figuren

2.1 Het concept van een ERP-systeem 8

3.1 Basiselementen van het datawarehouse 15
3.2 Inmon: bron van alle datamarts (top-down) 19
3.3 Kimball: unie van alle datamarts (bottom-up) 19
3.4 Dimensionaal gegevensmodel 22

4.1 Voorbeeld van een beslissingsboom: Koopt computer? 32

5.1 Boomstructuur 37

6.1 Artikelfiche in Lijn BOB 55
6.2 Klantenfiche in Lijn BOB 56
6.3 Ingaven van de facturen in Lijn BOB 57
6.4 Onderlinge relaties tussen de tabellen IART, COMPAN en IHISTO 57

7.1 Het sterschema van het verkoopsproces 62
7.2 ETL4ALL architectuur 63
7.3 Doelgegevens: huizen 65
7.4 Gegevensbron: veelhoeken 65
7.5 Gegevensbron: driehoeken 66
7.6 Transformaties voor het voorbeeld met huizen 66
7.7 Gegevensbronnen voor het voorbeeld met huizen 68
7.8 Grafische voorstelling van de metadata definities van IART, COMPAN en IHISTO in ETL4ALL 70
7.9 De procedure *vulDatamart* 73
7.10 De procedure *berekenWinst* 74
7.11 Het programma *maakDatamart* 75
7.12 De virtuele gegevens van *datamart* 75
7.13 De metadata definitie *klantgegevens* 77
7.14 De procedure *vulKlantGegevens* 80
7.15 De procedure *berekenRatios* 81

8.1 Sample 83
8.2 Explore 85
8.3 Modify 86
8.4 De oorspronkelijke verdeling van de variabele TotProd en ProdTrouw 86
8.5 De verdeling van het logaritme van de variabele TotProd en ProdTrouw 86
8.6 Model 87

8.7 Assess . . . 89
8.8 Visuele voorstelling van de beslissingsboom . . . 90
8.9 De eerste drie knopen van de beslissingsboom . . . 93

9.1 Integratie van een datamining model in een ERP-systeem . . . 97

A.1 De tabel IART . . . 101

B.1 De tabel COMPAN . . . 103

C.1 De tabel IHISTO . . . 105

E.1 De procedure *berekenWaarde* . . . 109
E.2 De procedure *berekenTrouwKlant* . . . 110
E.3 De procedure *berekenTrouwKlantVervolg* . . . 110
E.4 De procedure *berekenTrouwKlantVervolg2* . . . 111
E.5 De procedure *berekenProdWinJaar* . . . 111
E.6 De procedure *berekenSom* . . . 112

F.1 De variabelen LaatsteAankoop en TrouwKlant . . . 113
F.2 De variabelen JaarProd en JaarWinst . . . 113
F.3 De variabelen TotProd en ProdTrouw . . . 114
F.4 De variabelen TotWinst en WinstTrouw . . . 114

G.1 De top van de beslissingsboom . . . 115
G.2 De linkse vertakkingen van de beslissingsboom . . . 116
G.3 De middelste vertakkingen van de beslissingsboom . . . 116
G.4 De rechtse vertakkingen van de beslissingsboom . . . 117

Hoofdstuk 1

Probleemstelling

In dit inleidende hoofdstuk wordt de aanvankelijk vage probleemaanduiding omgezet tot een heldere en duidelijk omlijnde probleemstelling. Deze probleemstelling is tweeledig. Na de verkenning en verheldering van het praktijkprobleem in het eerste deel van dit hoofdstuk, wordt in het tweede deel het onderzoeksprobleem uitgewerkt. Tenslotte wordt de motivatie verwoord voor het gekozen onderwerp.

1.1 Verkenning en verheldering van het praktijkprobleem

1.1.1 Enterprise Resource Planning (ERP)

In de afgelopen jaren hebben ondernemingen in sterk toenemende mate geïnvesteerd in nieuwe informatiesystemen waarbij de integratie van systemen en de ondersteuning van zowel primaire als secundaire bedrijfsprocessen centraal staan. Veel ondernemingen beschikken daardoor tegenwoordig over een ERP(Enterprise Resource Planning)-pakket. Deze software ondersteunt niet alleen het volledige scala aan bedrijfsprocessen die zich binnen een organisatie afspelen, maar combineert dit tevens met een integrale aanpak van de informatisering van de diverse bedrijfsprocessen. Als gevolg hiervan herbergt het transactiesysteem van deze pakketten een grote hoeveelheid bedrijfsgegevens, die bovendien binnen één centrale database worden verzameld. De beschikbaarheid van bedrijfsinformatie op operationeel niveau wordt op die manier aanzienlijk vergroot. Daaruit ontstaat vervolgens vaak de verwachting dat de invoering van een ERP-pakket ook belangrijke voordelen kan hebben voor de voorziening van managementinformatie. Maar hoewel de verzamelde gegevens potentiële voordelen bieden voor het verbeteren van managementinformatie, wordt geen verbetering in de managementrapportage bewerkstelligd door het verzamelen van deze gegevens in een ERP-systeem (Bollen en Vluggen 2001). Het destilleren van managementinformatie uit een ERP-systeem lijkt in de praktijk nochtans een zeer relevante vraag. Uit een studie van Bothof en Götte (1998) blijkt immers dat de behoefte aan geïntegreerde managementinformatie inderdaad één van de voornaamste beoogde

doelstellingen van een ERP-implementatie is (Bollen en Vluggen 2001).

1.1.2 Beperkingen van een ERP-systeem

De beperkingen van een ERP-systeem kunnen als volgt worden samengevat (Bellomo 1999):

1. ERP-databases bevatten operationele gegevens terwijl voor het uitvoeren van analyses eerder behoefte is aan historische gegevens en indelingen van gegevens die niet direct aan het operationele proces zijn verbonden.

2. De ERP-database kan niet worden aangevuld met externe informatie. Voor beslissingen op tactisch en strategisch vlak zijn externe gegevens juist noodzakelijk zoals demografische gegevens en klantgegevens. In het algemeen geldt dat de inzet van informatietechnologie in organisaties vaak beperkt is gebleven tot het verzamelen en vastleggen van interne data.

3. De databasestructuur van ERP-software vertraagt de responstijd van ad hoc zoekopdrachten aanzienlijk.

De eerste twee genoemde beperkingen vloeien voort uit het feit dat ERP-systemen, operationele systemen zijn. Ze leggen de huidige stand van zaken vast. De derde beperking wordt enerzijds veroorzaakt door een gebrek aan flexibiliteit en functionaliteit, maar anderzijds ook doordat ERP-databases ontworpen zijn voor het optimaal vastleggen van transactiegegevens. Dit maakt ze echter ongeschikt voor het uitvoeren van multidimensionale analyses, waarbij gegevens worden gegroepeerd en geanalyseerd op basis van indelingen die vanuit het perspectief van de gebruiker interessant zijn (bijvoorbeeld per klant, per geografisch gebied) (Schroeck 1999). Het zijn juist deze analyses die voor het management van groot belang zijn. Voor een op het management afgestemde databasestructuur wordt daarom vaak gezocht naar een oplossing die naast de bestaande ERP-database functioneert (Bollen en Vluggen 2001).

1.1.3 Business Intelligence

Indien ondernemingen daadwerkelijk voordeel willen halen uit hun ERP-pakket voor de voorziening van managementinformatie, dienen ze het ontsluiten van de ERP-gegevensbron prioriteit te geven. Hiervoor kan men gebruik maken van analysesoftware, ook wel *Business Intelligence (BI)* toepassingen genoemd. BI wordt omschreven als het proces van verzamelen, analyseren en verspreiden van informatie met als doel een verbetering van de bedrijfsvoering te ondersteunen. Daarbij wordt vaak opgemerkt dat BI als doelstelling heeft het transformeren van

gegevens[1] in informatie[2] en deze informatie vervolgens te transformeren naar kennis[3]. De BI-systemen die worden gebruikt om deze kennis te genereren, vormen een belangrijk instrument waarmee de steeds maar toenemende snelheid en complexiteit van besluitvormingsprocessen binnen organisaties in belangrijke mate kunnen worden ondersteund. Ondernemingen die deze analysesoftware goed weten te benutten, kunnen rekenen op de volgende voordelen (Bollen en Vluggen 2001):

- Een kwaliteitsverbetering van informatie
- Een verbetering van de beschikbaarheid van informatie over applicaties heen
- Nieuwe functionaliteiten die het gebruikers mogelijk maken om snel en efficiënt tactische en strategische beslissingen te nemen

De basis voor de analyses die met BI-toepassingen worden verricht, wordt bepaald door de beschikbaarheid van operationele gegevens. Door de opkomst van ERP-systemen ontstond binnen veel organisaties een sterke vergroting van de hoeveelheid beschikbare data, die in één geïntegreerd systeem werden samengebracht. Daarmee kregen de mogelijkheden voor beslissingsondersteunende toepassingen een enorme impuls. Voor sommige BI-applicaties vormt het ERP-systeem een voldoende basis, maar voor andere toepassingen moeten aanvullende faciliteiten worden ingericht in de vorm van een datawarehouse. Een datawarehouse verzorgt het proces van het onttrekken van gegevens aan een ERP-database, de transformatie van deze gegevens en tot slot het laden van de getransformeerde gegevens in een aparte database (Eckerson 2000).

Een relatief nieuwe technologie, genaamd datamining, situeert zich binnen de groep van BI-systemen. Datamining is een techniek waarbij men op basis van een grote hoeveelheid beschikbare gegevens, op zoek gaat naar patronen zonder veel inzicht in de aard van de te onderzoeken problematiek. Met behulp van geavanceerde (statistische) technieken tracht men nieuwe relaties te ontdekken in de aanwezige gegevens om vervolgens te bepalen of de gevonden relatie op enige wijze kan worden benut (Bollen en Vluggen 2001).

1.2 Uitwerking van het onderzoeksprobleem

De uitwerking van het onderzoeksprobleem als tweede onderdeel van de probleemstelling, houdt in dat wordt nagegaan welke kennis alsnog ontbreekt om het gestelde praktijkprobleem afdoend

[1] Feiten of waarnemingen van fysieke verschijnselen of zakelijke transacties (O'Brien 1998)

[2] Gegevens die voor een eindgebruiker in hun context van betekenis zijn (O'Brien 1998)

[3] Georganiseerde en gecontextualiseerde informatie die gebruikt kan worden voor het produceren van nieuwe betekenissen en het genereren van nieuwe data (Wikipedia 2005)

te kunnen aanpakken. In een eerste stap wordt uit het praktijkprobleem een centrale onderzoeksvraag afgeleid. In de volgende stap wordt de werkwijze omschreven waarmee het antwoord zal worden gezocht op de centrale onderzoeksvraag.

1.2.1 Afleiding van de centrale onderzoeksvraag

In deze eindverhandeling trachten we een aantal eerder genoemde technologieën samen te brengen in een Business Intelligence context, namelijk ERP-systemen en datamining. Om deze integratie mogelijk te maken, zullen we gebruik maken van PMML of Predictive Model Markup Language. Dit is een vrij recent ontwikkelde standaard die voorziet in een eenvoudige en snelle manier om datamining modellen te definiëren en uit te wisselen tussen diverse toepassingen. Aan de hand van een concrete gevalstudie trachten we te onderzoeken of de PMML-standaard kan worden toegepast om datamining modellen te integreren in een ERP-systeem. We komen daarom tot volgende centrale onderzoeksvraag:

Kan een datamining model geïmplementeerd worden in een ERP-systeem met behulp van PMML?

Deze algemene onderzoeksvraag kan verder worden opgesplitst in twee deelvragen:

1. Op welke wijze kan een datamining model geïmplementeerd worden in een ERP-systeem door toepassing van PMML?

2. Is het integreren van een datamining model in een ERP-systeem door middel van PMML uitvoerbaar in de praktijk?

1.2.2 Strategie van het verkennend onderzoek

Voor het onderzoeken van het gestelde probleem, worden twee strategieën toegepast, namelijk een literatuurstudie en een gevalstudie. Eerst en vooral wordt een studie uitgevoerd van de bestaande literatuur om meer inzicht te bekomen in de voornaamste onderzoeksbegrippen. In dit theoretische gedeelte worden alle begrippen en technieken omschreven die bij het uitwerken van de gevalstudie aan bod komen. De literatuurstudie bestaat uit volgende onderdelen:

- Hoofdstuk 2: Enterprise Resource Planning
- Hoofdstuk 3: Datawarehousing
- Hoofdstuk 4: Datamining
- Hoofdstuk 5: Predictive Model Markup Language

Met behulp van een concrete gevalstudie trachten we nadien een antwoord te formuleren op de centrale onderzoeksvraag. De resultaten van deze studie worden gedetailleerd verwoord in volgende hoofdstukken:

- Hoofdstuk 6: Analyse van de gegevensstructuur in Lijn BOB
- Hoofdstuk 7: Constructie van een datamart met behulp van ETL4ALL
- Hoofdstuk 8: Ontwikkeling van een datamining model in PMML met behulp van SAS Enterprise Miner

Tenslotte worden in het laatste hoofdstuk conclusies geformuleerd met betrekking tot de centrale onderzoeksvraag op basis van de resultaten van de uitgewerkte gevalstudie.

1.3 Motivatie van het gekozen onderwerp

Mijn keuze voor dit onderwerp is gebaseerd op een aantal factoren. Eerst en vooral sloot dit onderwerp nauw aan bij mijn afstudeerrichting namelijk Beleidsinformatica. Begrippen zoals datawarehousing en datamining zijn zeer actueel in het huidige bedrijfsleven, en op deze manier hoopte ik dan ook de nodige kennis op te doen die mij een meerwaarde zouden opleveren in mijn latere professionele leven.

Daarnaast ben ik erg geïnteresseerd in het domein van Business Intelligence. Dit thema kwam reeds eerder aan bod bij een aantal opleidingsonderdelen waaronder Managementinformatiesystemen, KMO-informatiesystemen en Artificiële Intelligentie. Via deze eindverhandeling hoopte ik nog meer voeling te krijgen met dit onderwerp en er een bredere visie op na te houden.

Een andere factor die mijn keuze heeft bepaald, was de samenwerking met het bedrijf IKAN Software. Dit leek mij een unieke kans om ervaring op te doen met het bedrijfsleven. Tenslotte bood de gekozen onderzoeksstrategie de kans om zowel via de theorie als via de praktijk inzicht te bekomen in de verschillende onderzoeksbegrippen.

Deel I

Literatuurstudie

Hoofdstuk 2

Enterprise Resource Planning

In dit hoofdstuk wordt het concept Enterprise Resource Planning of ERP geïntroduceerd. Uitgaande van enkele definities, worden de basiskenmerken van ERP aangegeven. Nadien volgt een korte beschrijving van de geschiedenis van ERP en de voor- en nadelen van een ERP-implementatie. Tenslotte wordt in het laatste deel de toekomst van ERP toegelicht.

2.1 Wat is Enterprise Resource Planning?

2.1.1 Definitie

Het concept Enterprise Resource Planning (ERP) kan vanuit diverse perspectieven worden beschouwd. Enerzijds is ERP computersoftware. Anderzijds kan het ook worden aanzien als een middel voor het vastleggen van alle processen en gegevens van een organisatie en voor het creëren van een allesomvattende geïntegreerde structuur.

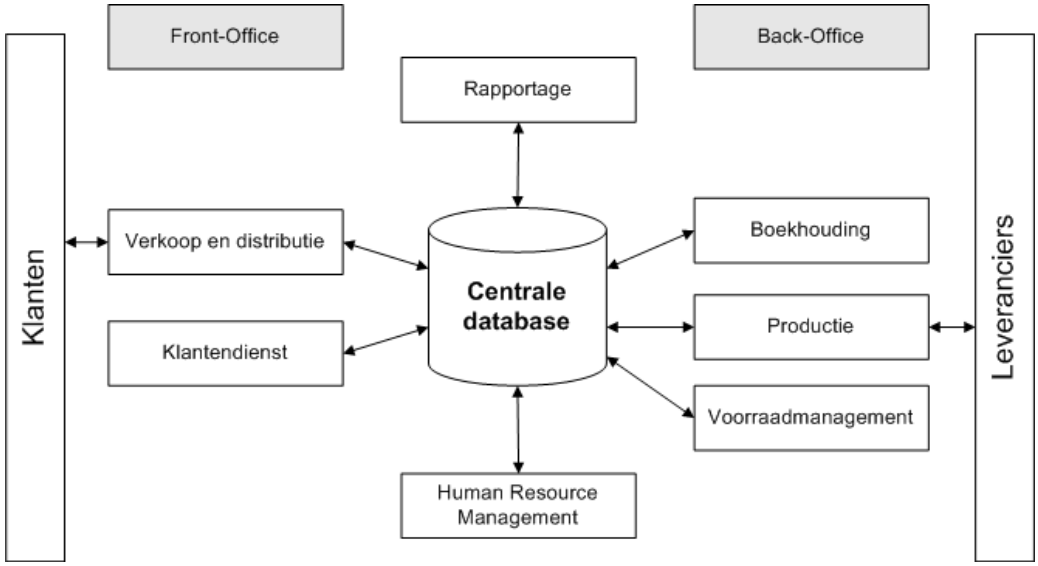
Er bestaan meerdere definities voor ERP die min of meer gelijkaardig zijn. Klaus (2000) definieert ERP als:

> „Totale softwareoplossing voor het integreren van alle processen en functies van een organisatie met als doel het presenteren van een holistisch beeld van de organisatie vanuit één enkele informatie- en informatietechnologie-architectuur."

Chase *et al.* (1998) verwijst naar ERP als:

> „Een computersysteem dat toepassingen voor boekhouding, verkoop, productie en andere functies van het bedrijf integreert. Deze integratie wordt bereikt door middel van één database die wordt gedeeld door alle toepassingen."

Beide definities geven duidelijk aan dat integratie centraal staat binnen ERP. Een ERP-systeem biedt de mogelijkheid om alle processen en functies van een bedrijf te integreren om op die manier een algemeen beeld te vormen van de organisatie. ERP beoogt een vloeiende integratie van alle gegevens die doorheen een bedrijf stromen, door gebruik te maken van één enkele database die toelaat dat verschillende afdelingen binnen een organisatie op efficiënte wijze gegevens uitwisselen en met elkaar communiceren (Davenport 1998). Als gevolg van deze benadering hebben alle afdelingen toegang tot dezelfde data en wordt redundantie van gegevens geëlimineerd (figuur 2.1). Bovendien verbetert de kwaliteit van gegevens. Voorheen gebruikten bedrijven vaak verschillende versies van data hetgeen aanleiding gaf tot foutieve informatie en slechte beslissingen (Yen *et al.* 2002).



Figuur 2.1: Het concept van een ERP-systeem

2.1.2 Evolutie van ERP

Gedurende de periode 1960-1970 ontwikkelden en implementeerden organisaties voornamelijk software voor voorraadcontrole op basis van traditionele voorraadconcepten. In 1970 werd Material Requirements Planning (MRP) geïntroduceerd. Deze methode splitst een afgewerkt product op in al haar samenstellende componenten uitgaande van de vraag naar het eindproduct in het Master Production Schedule (MPS). Het MPS is een tijdschema waarin wordt vastgelegd wanneer de productie van het eindproduct dient te gebeuren en hoeveel eenheden moeten worden geproduceerd (Chase *et al.* 1998). Hoewel MRP een relatief eenvoudige werkwijze was, werd de hoeveelheid gegevens in een realistische situatie erg omslachtig. Bovendien beperkte deze methode zich tot het plannen van de vereisten van ruwe materialen. Als reactie op deze

beperkingen ontstond rond 1980 Manufacturing Resource Planning (MRPII). MRPII legde de nadruk op het optimaliseren van het volledige productieproces door het synchroniseren van de materialen met de productievereisten. Hoewel MRPII oorspronkelijk slechts een uitbreiding was van MRP en enkel werkvloeractiviteiten en distributiemanagement ondersteunde, werden de systemen verder uitgebreid naar verschillende gebieden zoals financieel management, human resources, engineering, projectmanagement, ... Dit leidde tot het ontstaan van de eerste ERP-systemen in 1990. Gebaseerd op de technologische fundamenten van MRP en MRPII, voorzagen deze systemen in de cross-functionele coördinatie en integratie van het productieproces. ERP-systemen omvatten, in tegenstelling tot hun voorgangers, het volledige scala van bedrijfsprocessen en leveren bovendien toegang, transparantie en consistentie doorheen de gehele organisatie (Rashid *et al.* 2002).

Sinds 1970 is het idee reeds aanwezig van een geïntegreerd informatiesysteem dat alle functies en processen van een bedrijf omvat. In deze periode werden informatiesystemen echter zelden geïntegreerd en wanneer een nieuwe toepassing werd toegevoegd, werd ze slechts in geringe mate gekoppeld aan de bestaande systemen. Deze praktijk leidde na enkele jaren tot het ontstaan van een lappendeken van redundante systemen. Deze gefragmenteerde systeemarchitectuur vertoonde bovendien een aantal moeilijkheden. De gegevenskwaliteit was erg laag doordat de verschillende systemen niet gelijktijdig werden bijgewerkt. De onderhoudskost van deze systemen nam zeer snel toe waardoor er geen middelen overbleven om nieuwe systemen te ontwikkelen. In deze context ging met de introductie van ERP-pakketten, die een gestroomlijnde integratie beloofden, een droom in vervulling (Eskilsson *et al.* 2003).

2.2 ERP-systemen in de organisatie

Een algemeen misleidende perceptie van een ERP-implementatie is dat na de implementatie de functies van een organisatie onmiddellijk zullen verbeteren. De mate waarin de hoge verwachtingen (zoals het besparen van kosten en het verbeteren van processen) worden ingelost, is erg afhankelijk van hoe goed het gekozen ERP-systeem is afgestemd op de behoeften, de cultuur, de strategie en de structuur van de organisatie.

Organisaties kiezen doorgaans voor het implementeren van een ERP-systeem omwille van volgende voordelen (Rashid *et al.* 2002):

- *Toegang tot betrouwbare informatie*
  Het gebruik van een gemeenschappelijke database leidt tot consistente, accurate en betere gegevens.

- *Eliminatie van redundante gegevens en operaties*
  Aangezien alle afdelingen binnen een organisatie toegang hebben tot dezelfde gegevens in de centrale database, wordt het invoeren van dubbele gegevens en het uitvoeren van dubbele operaties vermeden.

- *Vermindering van leverings- en cyclustijden*
  ERP-systemen verkorten de tijd die nodig is voor het opzoeken en rapporteren van gegevens uit de centrale database.

- *Kostenreductie*
  Een kostenbesparing wordt voornamelijk gerealiseerd door het uitsparen van tijd en het beschikken over betere informatie voor het nemen van beslissingen.

- *Verbeterde schaalbaarheid*
  De huidige ERP-systemen beschikken meestal over een modulair opgebouwde structuur waardoor ze gemakkelijk uitbreidbaar zijn.

- *Verbeterd onderhoud*
  Vaak worden onderhoudscontracten op lange termijn afgesloten met de ERP-leverancier.

Om deze voordelen te bekomen, moeten ondernemingen een aantal problemen en moeilijkheden overwinnen, namelijk (Rashid *et al.* 2002):

- Tijdrovende implementatie
- Hoge kostprijs
- Conformiteit van het ERP-systeem met de processen, de cultuur en de strategische doelen van de organisatie
- Afhankelijkheid van één systeem en één leverancier vanwege de bedrijfsbrede toepassing van het ERP-systeem
- Grote impact van de installatie en ingebruikname (implementatie) van het ERP-systeem op de organisatie

2.3 Commerciële ERP-pakketten

ERP-systemen zijn vaak commerciële softwarepakketten die gekocht worden van softwareleveranciers, en bijgevolg niet perfect voldoen aan de behoeften van een organisatie. In de praktijk blijkt dat ERP-software slechts aan 70 procent van de behoeften van een organisatie kan beantwoorden. Een organisatie heeft bijgevolg drie mogelijkheden: Het kan haar processen

afstemmen op het pakket, het kan het pakket aanpassen aan haar processen of het kan beslissen om de overige 30 procent te verwaarlozen. Vaak wordt gekozen om de eigen bedrijfsprocessen aan te passen aan het nieuwe systeem. Dit is te wijten aan het feit dat het op maat maken van ERP-pakketten een aantal negatieve gevolgen heeft. Maatwerk verhoogt over het algemeen de implementatiekost, de implementatieduur, de afhankelijkheid van de leverancier en de onderhoudskost (Eskilsson *et al.* 2003).

Enkele voorname ERP-softwareleveranciers zijn SAP, Oracle, PeopleSoft en Baan. Zij richtten zich aanvankelijk op de grotere ondernemingen die complexe toepassingen vereisen. De laatste jaren kan men echter een verschuiving vaststellen naar de markt van de kleine en middelgrote ondernemingen (KMO's) waarbij men voornamelijk streeft naar goedkope, vooraf geconfigureerde en eenvoudig te installeren oplossingen (Rashid *et al.* 2002).

2.4 Toekomst van ERP

Het meerendeel van de bedrijfsprocessen worden zeer goed ondersteund door ERP-systemen. Toch bieden deze systemen niet de beste oplossing voor processen die minder gegevensintensief zijn zoals het management van de voorraadketen, klantenmanagement en marketing. Gelukkig hebben ontwikkelaars van ERP-systemen deze behoefte tijdig ingezien en bieden ze steeds meer oplossingen aan die deze bedrijfsprocessen ondersteunen. Zo bestaan er uitbreidingen naar Supply Chain Management (SCM) en Customer Relationship Management (CRM) om de *front office* activiteiten (verkoop, marketing, klantenservice) te koppelen aan de *back office* activiteiten (productie, boekhouding, human resources) met als doel het bekomen van competitief voordeel (Eskilsson *et al.* 2003). Supply Chain Management kan worden omschreven als de activiteiten die erop gericht zijn om alle partijen in de voorraadketen zodanig te laten samenwerken dat de consument optimaal wordt bediend en waarbij de gezamenlijke kosten zo laag mogelijk zijn (Van der Veen en Robben 1997). Customer Relationship Management richt zich op het beheren van alle contacten van een bedrijf met bestaande en potentiële klanten. CRM maakt gebruik van informatiesystemen om alle bedrijfsprocessen te integreren rond de interacties van een organisatie met haar klanten (Laudon en Laudon 2004).

Hoofdstuk 3

Datawarehousing

Dit hoofdstuk tracht een algemeen overzicht te bieden van het begrip datawarehousing. Het eerste deel toont de noodzaak aan van datawarehousing, geeft een algemene definitie en de voornaamste doelstellingen. Vervolgens wordt dieper ingegaan op de architectuur en andere aanverwante begrippen.

3.1 Wat is datawarehousing?

3.1.1 Noodzaak van datawarehousing

Een datawarehouse is een architectuur van informatiesystemen die gebruikers voorziet van beslissingsondersteunende informatie, die in traditionele operationele gegevensbestanden moeilijk te benaderen of weer te geven is. Levering van geïntegreerde, strategische managementinformatie vanuit operationele systemen stuit op volgende hindernissen (Snijders en Van Reijswoud 2000), (Laudon en Laudon 2004):

- **Geen stabiele omgeving voor analyses**
  Gegevens in operationele systemen wijzigen voortdurend. Kenmerkend voor strategische analyse is dat een vraag een volgende vraag oproept. Als deze vragen aan een operationeel systeem worden gesteld, bestaat de kans dat bij de vervolgvraag de situatie weer veranderd is en de cijfers niet meer overeenkomen met de initiële vraag.

- **Beperkte historiek**
  Operationele systemen zijn bedoeld voor operationeel/tactisch gebruik. Een langdurige historiek bijhouden biedt voor dergelijk gebruik geen voordelen; het zou alleen de prestaties negatief beïnvloeden. Voor strategische analyses zijn historische gegevens daarentegen noodzakelijk.

- **Beperkingen met betrekking tot prestaties**
  Een operationeel systeem dient een gegarandeerd prestatieniveau te bieden. Ad hoc

zoekopdrachten hebben de neiging een systeem langdurig zwaar te belasten, met als gevolg dat het totale prestatieniveau lager ligt.

- **Vergrendelen of *locking* van gegevens**
  Vergrendelingsmechanismen worden gebruikt in databases om inconsistentie te voorkomen. Hoewel dit voor individuele transacties zelden een probleem vormt, kan een grote zoekopdracht als gevolg van *locking* door een andere transactie, lang worden tegengehouden.

- **Vervuiling**
  Het is een praktijkgegeven dat elke database, groter dan enkele megabytes, enige mate van bevuiling bevat. Het gebruik van vervuilde gegevens voor analyses kan het resultaat zwaar vertekenen.

- **Gefragmenteerde gegevens**
  De gegevens die nodig zijn voor het nemen van beslissingen, zijn vaak verspreid over meerdere operationele databases of externe bronnen waardoor het gevaar ontstaat dat verschillende gebruikers beslissingen nemen op basis van onvolledige gegevens.

- **Verschillende technische infrastructuren**
  In de meeste organisaties wordt een verscheidenheid aan hard- en softwareplatforms gebruikt. Dit veroorzaakt veelal een probleem bij het vergelijken van gegevens vanuit verschillende operationele systemen.

- **Ongelijksoortige datastructuren**
  Gegevens uit verschillende applicaties zijn vaak niet zonder ingewikkelde transformaties te vergelijken.

- **Moeilijke toegang tot gegevens**
  Gegevens uit operationele databases zijn doorgaans niet eenvoudig te raadplegen door gebruikers zonder enige computerexpertise.

3.1.2 Definitie

Ons gehele bestaan is een proces van verzamelen, analyseren en begrijpen van informatie. Informatie is de kern van alles om ons heen. Datawarehousing is ontstaan als een erkenning van de waarde en de rol van informatie. Een datawarehouse is in feite een middel voor het strategisch gebruik van gegevens. Het is een platform met geïntegreerde, kwalitatief hoogwaardige gegevens ter ondersteuning van vele beslissingsondersteunende toepassingen en processen binnen een onderneming.

Een formele definitie van het datawarehouse komt van Bill Inmon (2002) :

> „Een datawarehouse is een onderwerp-georiënteerde, geïntegreerde, niet vluchtige en tijdsafhankelijke verzameling van gegevens ter ondersteuning van beslissingen op managementniveau."

Deze definitie weerspiegelt de voornaamste doelstelling van een datawarehouse namelijk het bieden van ondersteuning. De vier eigenschappen die worden vernoemd, onderscheiden een datawarehouse van operationele systemen (Bakker *et al.* 2001):

- Een datawarehouse is onderwerp-georiënteerd aangezien het georganiseerd is rondom de belangrijkste onderwerpen van de informatiebehoeften van een organisatie. Denk hierbij bijvoorbeeld aan informatie over klanten of over producten. Dit in tegenstelling tot de operationele bronsystemen die zijn afgeleid van functies en processen in een organisatie zoals bijvoorbeeld het versturen van facturen.

- Het tweede kenmerk, namelijk integratie, is een zeer belangrijke eigenschap. Een datawarehouse verzamelt gegevens die afkomstig zijn van diverse bronnen verspreid over de organisatie. De verschillende bestaande systemen die als bron dienen voor het datawarehouse kunnen allen een eigen notatiewijze en definitie hanteren voor de opgeslagen data. Binnen een datawarehouse worden de definitie en notatiewijze van deze data op een eenduidige manier vastgelegd, waardoor misverstanden omtrent de interpretatie van de gegevens worden voorkomen.

- Een datawarehouse is een tijdsafhankelijke en niet vluchtige verzameling van gegevens. Alle gegevens in het datawarehouse zijn voorzien van een soort tijdsstempel. De in het datawarehouse opgeslagen gegevens worden na foutloze overname en eventueel noodzakelijke correcties doorgaans niet meer bijgewerkt of veranderd. Mocht er toch een verandering moeten worden doorgevoerd, dan krijgt de aanpassing een tijdsstempel. Door dit tijdsstempel kan altijd de toestand van vóór de verandering worden achterhaald en gaat de historiek dus niet verloren. Doordat in een datawarehouse de historiek van gegevens wordt bewaard, zijn ook vergelijkingen met voorgaande periodes mogelijk (trendanalyse).

3.1.3 Doelstellingen

Het ontwerpen, ontwikkelen, beheren en gebruiken van een datawarehouse is een tijdrovend en vaak kostbaar proces. Een geslaagde datawarehouse-implementatie kan echter belangrijke, concrete voordelen realiseren. Volgens Kimball en Ross (2002) zijn de fundamentele doelstellingen van een datawarehouse:

1. Eenvoudige toegang verschaffen tot de gegevens van een organisatie

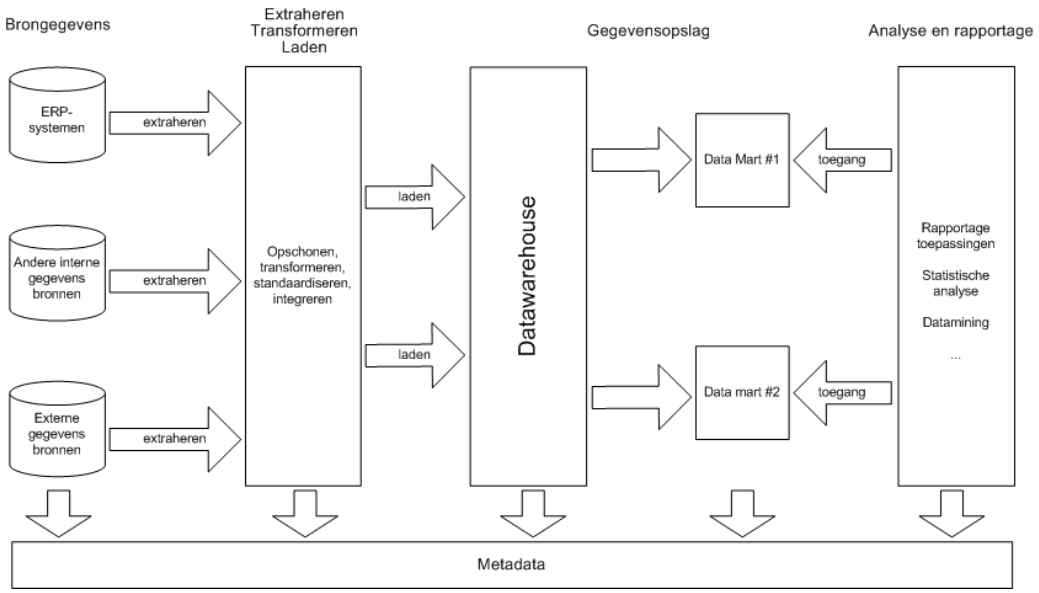
2. De gegevens van een organisatie op een consistente manier voorstellen
3. Flexibel zijn voor toekomstige veranderingen
4. Een veilige opslag voorzien van gegevens
5. Een basis vormen voor het nemen van betere bedrijfsbeslissingen
6. Geaccepteerd worden door de gehele organisatie

3.2 Datawarehouse-architectuur

Het datawarehouse is een omgeving en geen product. Het is een combinatie van technologieën en componenten gericht op een effectieve integratie van operationele databases in een omgeving die een strategisch gebruik van gegevens mogelijk maakt (Berson en Smith 1997). In deze paragraaf wordt aandacht besteed aan de architectuur van het datawarehouse en de componenten waaruit het is opgebouwd.

3.2.1 Basiselementen

Figuur 3.1 illustreert de vijf basiselementen van een datawarehouse-omgeving.



Figuur 3.1: Basiselementen van het datawarehouse

3.2.1.1 Brongegevens

De brongegevens behoren tot de zogenoemde *back-end* van het datawarehouse. Het *front-end* van het datawarehouse wordt gevormd door de rapportage- en analysetoepassingen. De brongegevens zijn afkomstig van de informatiesystemen die binnen een organisatie in gebruik zijn. Hieronder vallen zowel interne als externe gegevensbronnen, maar ook gegevens die niet zijn opgeslagen in geautomatiseerde informatiesystemen. De bronsystemen leveren gegevens aan het datawarehouse. Van groot belang voor deze gegevens is daarom dat ze compleet, betrouwbaar en toegankelijk dienen te zijn. Dit omdat het eindresultaat aan de kant van de eindgebruiker sterk afhankelijk is van de kwaliteit van de gegevens waarmee het datawarehouse wordt gevoed. Als hetgeen zich in de bronbestanden bevindt van slechte kwaliteit is, zullen de resultaten van de analyse ook een vertekend beeld opleveren. Dit wordt ook wel het *garbage in = garbage out*-principe genoemd (Bakker *et al.* 2001).

3.2.1.2 Extraheren - Transformeren - Laden (ETL)

De eerste stap in het ETL-proces bestaat uit het extraheren van gegevens in de datawarehouse-omgeving. Extraheren betekent het lezen en begrijpen van de brongegevens en vervolgens het kopiëren van de vereiste gegevens. Deze moeten nadien in een vorm worden gegoten die geschikt is voor de toepassingen die informatie uit het datawarehouse onttrekken (Berson en Smith 1997). De volgende stap in het ETL-proces is dan ook het uitvoeren van een aantal transformaties waaronder het opschonen van data ter verbetering van de kwaliteit, het combineren van data uit verschillende bronnen, ... Uiteindelijk worden de getransformeerde gegevens naar het datawarehouse getransporteerd (Kimball en Ross 2002).

Het ETL-proces is het meest kritieke en tijdrovende gedeelte van een datawarehouse-project. Ongeveer 60 tot 70 procent van het ontwikkellingstraject wordt door deze complexe, maar zeer belangrijke basisprocessen in beslag genomen. Door gebruik te maken van een ETL-toepassing kunnen de risico's van een datawarehouse-project aanzienlijk worden gereduceerd. Deze applicaties zijn specifiek ontwikkeld voor het extraheren van gegevens uit de bronsystemen, om deze vervolgens met transformatieregels te bewerken en in het datawarehouse te laden (Kamst en Coppens 1999).

3.2.1.3 Gegevensopslag

Met de term datawarehouse wordt in deze context het onderdeel bedoeld waarin de geïntegreerde gegevens over verschillende aandachtsgebieden van de organisatie daadwerkelijk, fysiek worden opgeslagen. Het voordeel van het creëren van een afzondelijke database, zoals hier het geval is, is dat tijdens het uitvoeren van een analyse in het datawarehouse het operationele systeem

niet wordt belast (Bakker *et al.* 2001).

De data in het datawarehouse dienen te voldoen aan een aantal kenmerken. Zij dienen accuraat, consistent en compleet te zijn. Accuraat houdt in dat de gegevens geschoond zijn en dat de relaties met andere gegevens geldig zijn. Consistentie van data doelt op het aanwezig zijn van alle noodzakelijke gegevens die verband houden met een bepaalde individuele transactie. De compleetheid van gegevens heeft betrekking op het aanwezig zijn in het systeem van de door de gebruiker verwachtte gegevens (Bakker *et al.* 2001).

Vanuit het datawarehouse kunnen zogenaamde datamarts worden afgeleid. In datamarts worden gegevens opgeslagen met betrekking tot een bepaald bedrijfsproces, onderwerp of afdeling. Een datamart bevat dus slechts een deel van de gegevens die in een organisatie worden verzameld. Ze zijn gericht op een specifiek aandachtsgebied ter ondersteuning van een individuele afdeling of divisie van een organisatie. Er zijn twee verschillende vormen van een datamart. Indien een datamart wordt gevuld met informatie afkomstig uit het centrale datawarehouse, wordt deze vorm van een datamart ook wel een afhankelijke datamart genoemd. Op het moment dat een datamart rechtstreeks wordt gevuld met gegevens uit de diverse operationele bronsystemen, wordt er gesproken van een onafhankelijke datamart. Ook combinaties van beide vormen zijn mogelijk (Bakker *et al.* 2001).

3.2.1.4 Analyse en rapportage

De analyse- en rapportagetoepassingen dienen ter ondersteuning van de eindgebruiker. Ze vormen de presentatielaag van het datawarehouse en worden daarom ook wel *front-end* applicaties genoemd. Met behulp van deze toepassingen krijgen de gebruikers op een eenvoudige manier toegang tot het datawarehouse (Bakker *et al.* 2001).

3.2.1.5 Metagegevens

Metagegevens zijn zeer belangrijk voor datawarehousing. Het zijn gegevens over gegevens die het datawarehouse beschrijven. Ze vormen als het ware de lijm die de andere onderdelen in de datawarehouse-architectuur bij elkaar houdt. In de metadata worden (eenduidige) definities vastgelegd van de in het datawarehouse opgeslagen gegevens. Hierbij wordt onder andere aangegeven uit welke bronsystemen de gegevens afkomstig zijn, hoe ze zijn vertaald uit de bronsystemen naar het datawarehouse en wat de veranderingen in de gegevensdefinities doorheen de tijd zijn. Uit alle onderdelen van het datawarehouse wordt informatie vastgelegd in de metadata (Bakker *et al.* 2001). Men kan volgend onderscheid maken (Berson en Smith 1997):

- **Technische metagegevens** bevatten informatie over datawarehouse-gegevens ten behoeve van datawarehouse-ontwerpers en beheerders om te gebruiken bij het uitvoeren

van ontwikkelings- en beheertaken. Voorbeelden zijn: informatie over gegevensbronnen, transformatiebeschrijvingen, regels die worden gebruikt bij het opschonen van data

- **Bedrijfsmatige metagegevens** bevatten informatie die een basis vormen waarop gebruikers zich een gemakkelijk te interpreteren beeld kunnen vormen van de gegevens die in het datawarehouse aanwezig zijn.

Metagegevens behoren te worden verzameld op het moment dat het datawarehouse wordt ontworpen en gebouwd. Om de overzichtelijkheid te bewaren, worden ze geïntegreerd en centraal opgeslagen. Dit wordt bereikt door gebruik te maken van een zogeheten metadatabewaarplaats. Dit is meestal een aparte database waarin de metadata worden opgeslagen en beheerd. Op deze manier wordt gegevensredundantie en inconsistentie vermeden (Berson en Smith 1997).

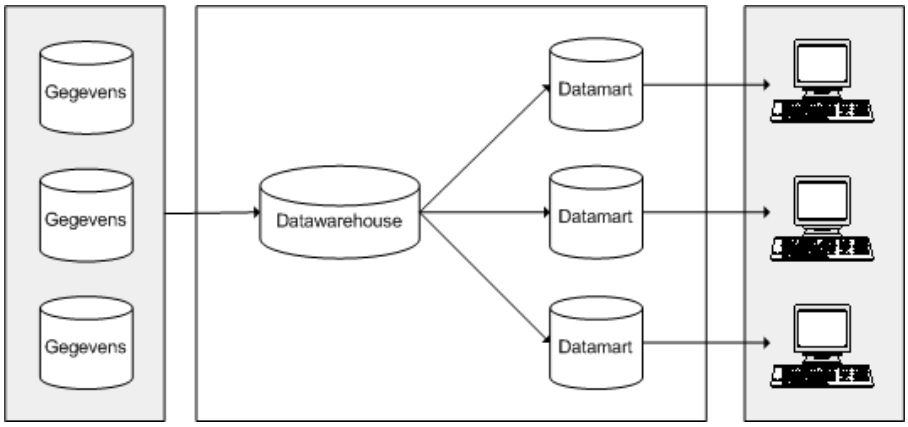
3.2.2 Top-down versus bottom-up benadering

Het concept datamart veroorzaakt heel wat verwarring. Er bestaat geen eenduidigheid over de waarde, de rol en de betekenis van een datamart. De voornaamste reden hiervoor is het bestaan van verschillende benaderingen voor het bouwen van een datawarehouse. Deze benaderingen zijn gebaseerd op de ideologieën van Ralph Kimball en Bill Inmon - twee experts in het kennisgebied datawarehousing. Bill Inmon is de grondlegger van het begrip datawarehousing en Ralph Kimball is de pionier van het dimensionaal modelleren (zie paragraaf 3.3).

De oorspronkelijke definitie voor een datamart komt van Bill Inmon. Hij definieert een datamart als een opslageenheid die ondergeschikt is aan een datawarehouse met geïntegreerde gegevens en die verwijst naar een deel van de gegevens (vaak een onderwerpgebied genoemd) dat specifiek gecreëerd is voor een groep gebruikers (Inmon 2002). Een andere definitie wordt gegeven door Ralph Kimball. Hij definieert een datamart als een logisch onderdeel van een datawarehouse dat wordt georganiseerd rond één bedrijfsproces of rond een groep van gerelateerde bedrijfsprocessen (Kimball *et al.* 1998).

Inmon beschouwt een datawarehouse als de bron van alle datamarts. Volgens zijn benadering wordt eerst een volledig datawarehouse geconstrueerd en vervolgens één of meer afhankelijke datamarts (top-down). Kimball's benadering daarentegen beschouwt een datawarehouse als een unie van datamarts. Daarbij ontwikkelt men eerst onafhankelijke datamarts en construeert men vervolgens van daaruit het datawarehouse (bottom-up). Beide opvattingen worden achtereenvolgens weergegeven in figuur 3.2 en 3.3.

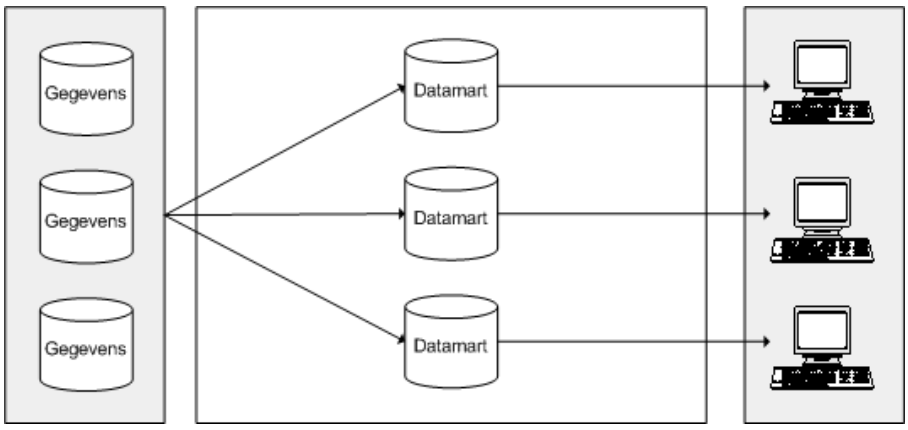
De methode van Inmon biedt als voordeel dat de bedrijfsgegevens in het datawarehouse uniform en consistent worden vastgelegd. Bovendien is slechts één centraal laadproces vanuit de operationele systemen noodzakelijk. De datawarehouse-omgeving wordt eenduidig opgezet, met



Figuur 3.2: Inmon: bron van alle datamarts (top-down)

standaard methoden en procedures voor het ontwerpen en ontwikkelen van gegevensextracties en transformaties. Het datawarehouse is opgezet naar een onderwerp-georiënteerd informatiemodel over de hele breedte van de organisatie. Problemen omtrent gegevensintegratie worden daardoor geminimaliseerd. Deze benadering opent de optie om de onderliggende datamarts te construeren vanuit het gegevensmodel van het datawarehouse (Kamst 2000).

Er zijn echter ook nadelen verbonden aan de top-down benadering van Inmon. Zo leert de ervaring dat top-down projecten vaak leiden tot zeer lange doorlooptijden, hoge implementatiekosten en een zwakke eindgebruikerfunctionaliteit. Verder blijkt ook dat de gewenste informatie te lang op zich laat wachten waardoor de informatiebehoefte tijdens de ontwikkeling meermaals verandert. De ontwikkelingscyclus zal bijgevolg altijd achter de feiten aanlopen (Kamst 2000).



Figuur 3.3: Kimball: unie van alle datamarts (bottom-up)

Omwille van bovenstaande nadelen kiest men vaak voor Kimball's methode, waarmee op eenvoudige en snelle wijze enkele datamarts kunnen worden gerealiseerd. Het nadeel van deze

keuze is dat de overzichtelijkheid en de flexibiliteit afnemen bij latere veranderingen. Naarmate het aantal datamarts toeneemt, neemt het overzicht af. Er zijn meer laadprocessen nodig en daardoor wordt het steeds complexer om de gegevens tussen de datamarts onderling consistent te houden. Dit laatste is noodzakelijk om te voorkomen dat de samenhang zoek raakt. Ook bij een wijziging in een datamart moet op consistentie worden gecontroleerd. Een wijziging in de ene datamart kan leiden tot aanpassingen in anderen, vooral wanneer gegevens in meerdere datamarts voorkomen. De flexibiliteit om datamarts te wijzigen neemt daardoor af (Kamst 2000).

In deze eindverhandeling volgen we de methode van Ralph Kimball omdat deze ons toelaat om snel en eenvoudig een datamart te ontwikkelen. Aangezien we in de gevalstudie slechts één datamart zullen construeren, moeten we geen rekening houden met de voornaamste nadelen van de benadering van Kimball en biedt de methode van Inmon geen bijkomende voordelen.

3.3 Dimensionaal gegevensmodel

De schemamethodologie die steeds vaker voor datawarehousing wordt gebruikt, is het dimensionaal gegevensmodel. In deze paragraaf geven we een kort overzicht van deze methode.

3.3.1 Algemeen

Volgens Kimball (1998) verschilt een datawarehouse-omgeving aanzienlijk van de operationele omgeving en zijn de technieken die worden toegepast voor het ontwerpen van operationele databases niet geschikt voor het ontwerpen van datawarehouses. Omwille hiervan, stelt Kimball een nieuwe techniek voor die specifiek gericht is op het ontwerpen van datawarehouses en datamarts, namelijk dimensionaal modelleren. Deze methode werd ontwikkeld op basis van observaties uit de praktijk, en werd nooit empirisch getest. Toch blijkt deze methode erg succesvol te zijn en wordt ze algemeen beschouwd als voornaamste benadering voor het ontwerpen van datawarehouses en datamarts. Het dimensionaal gegevensmodel heeft bovendien bijgedragen tot de discipline van data modellering en database-ontwerp (Moody en Kortink 2000).

3.3.2 Het doel van dimensionaal modelleren

Er bestaan twee grote verschillen tussen operationele databases en datawarehouses:

- Toegang van de eindgebruiker: In een datawarehouse-omgeving voeren eindgebruikers rechtstreeks query's uit op de databasestructuur. In een operationele omgeving, hebben gebruikers enkel toegang tot de database via een toepassing. In een traditionele toepassing, is de structuur van de database onzichtbaar voor de eindgebruiker.

- Enkel lezen: Datawarehouses kunnen enkel worden geraadpleegd - gebruikers kunnen gegevens opvragen en analyseren, maar niet wijzigen.

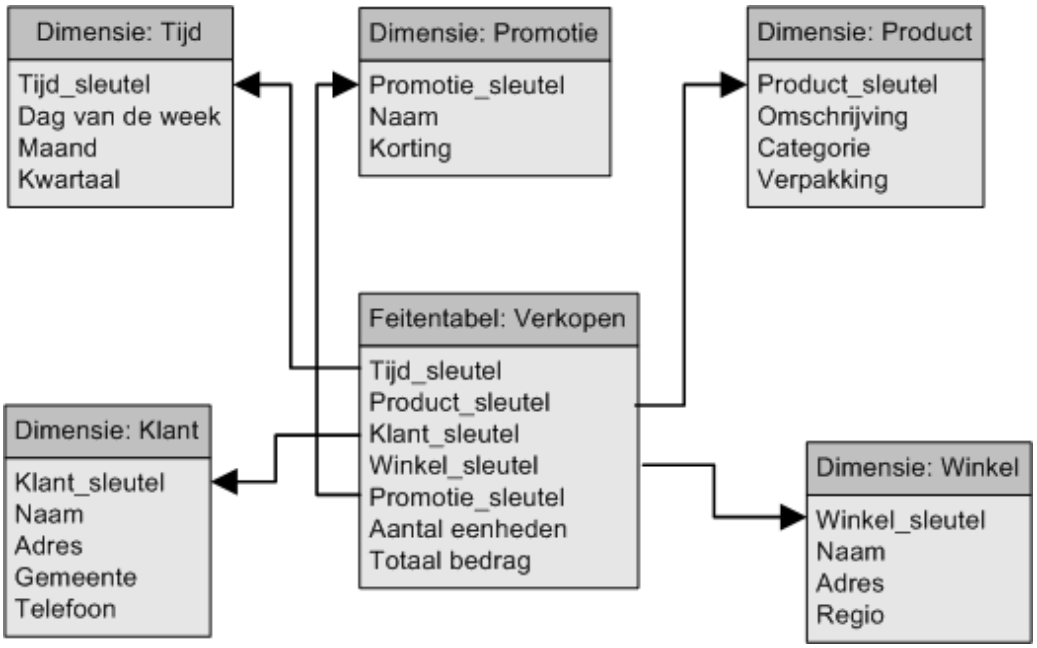
Het probleem bij traditionele database-ontwerptechnieken is dat ze vaak resulteren in databasestructuren die te complex zijn om door de eindgebruiker te worden begrepen en gebruikt. Een typische operationele database bestaat uit honderden tabellen die verbonden zijn door een complex web van relaties. Dit vormt geen probleem voor transactieverwerkende systemen omdat de complexiteit van de databasestructuur verborgen blijft voor de gebruiker. De voornaamste oorzaak voor de complexiteit van operationele databases is het toepassen van normalisatie. Normalisatie vermenigvuldigt het aantal vereiste tabellen, aangezien functioneel afhankelijke attributen worden opsplitst in afzonderlijke tabellen. De doelstelling van normalisatie is het minimaliseren van gegevensredundantie. Redundantie betekent concreet het opslaan van eenzelfde gegeven op meerdere locaties in de database. Wanneer dit gegeven nadien wordt aangepast, kan het aanleiding geven tot inconsistentie als wijzigingen niet overal worden doorgevoerd. Redundantie vormt een minder belangrijk probleem in een datawarehouse-omgeving omdat gegevens niet kunnen worden gewijzigd door de eindgebruikers (Moody en Kortink 2000).

Het objectief van dimensionaal modelleren is het produceren van databasestructuren die eindgebruikers eenvoudig kunnen begrijpen en gebruiken. Een tweede doelstelling is het maximaliseren van de efficiëntie van query's. Deze doelen worden voornamelijk bereikt door het minimaliseren van het aantal tabellen en relaties tussen de verschillende tabellen. Dit reduceert de complexiteit van de database (Moody en Kortink 2000).

3.3.3 Het sterschema

De basis van een dimensionaal gegevensmodel wordt gevormd door het sterschema. Een sterschema is een relationeel schema dat gerangschikt is rond een centrale tabel (feitentabel), gekoppeld aan enkele kleinere tabellen (dimensietabellen) (Berson en Smith 1997). De feitentabel bevat ruwe, numerieke metingen die een relevant zakelijk feit weergeven, bijvoorbeeld: de prijs van verkochte producten of het aantal verkochte producten. De primaire sleutel[1] van de feitentabel wordt gevormd door het samenstellen van de primaire sleutels van de dimensies waarmee de tabel is verbonden. Dimensietabellen bevatten attributen die gecorreleerd zijn en die de kenmerken omschrijven van een tastbaar feit, bijvoorbeeld: product of klant (Kimball *et al.* 1998). Deze tabellen bevatten een niet-samengestelde primaire sleutel. Tussen een dimensie en de feitentabel bestaat een één-op-veel relatie. Elke rij in de feitentabel verwijst naar precies één specifieke dimensie. Een dimensie daarentegen, kan in meerdere rijen van de feitentabel voorkomen. Een concreet voorbeeld van een sterschema wordt getoond in figuur 3.4.

[1] De primaire sleutel van een relationele tabel identificeert op unieke wijze elke rij in een databasetabel.



Figuur 3.4: Dimensionaal gegevensmodel

3.3.4 Voordelen van dimensionaal modelleren

Dimensionaal modelleren biedt drie voordelen ten opzichte van traditionele database ontwerptechnieken (Kimball en Ross 2002):

1. Het dimensionaal gegevensmodel is eenvoudig en symmetrisch. Hierdoor kunnen gebruikers het model gemakkelijk begrijpen en navigeren. Bovendien is het model zeer herkenbaar voor de eindgebruikers in een onderneming.

2. De eenvoud van het dimensionaal model biedt bovendien een aantal prestatievoordelen. Gegevens kunnen met behulp van eenvoudige query's worden opgevraagd.

3. Tenslotte kan men dimensionale modellen vloeiend uitbreiden om te beantwoorden aan veranderende gebruikersbehoeften.

3.3.5 Het vier-stappen-model voor het ontwerpen van een dimensionaal model

De structuur van een datamart wordt bepaald aan de hand van een dimensionaal gegevensmodel. Voor het ontwerpen van een dimensionaal schema, ontwikkelde Ralph Kimball een methodologie die bestaat uit 4 stappen (Kimball *et al.* 1998), (Kimball en Ross 2002):

Stap 1: Het onderwerp van een bepaalde datamart selecteren

In de eerste stap wordt het bedrijfsproces gekozen dat gemodelleerd zal worden. Voorbeel-

den van dergelijke processen zijn: het aankopen van grondstoffen, bestellingen, leveringen, facturatie, voorraadbeheer, ...

Stap 2: Het aangeven van de betekenis van de feitentabel

Na het bepalen van het onderwerp, moet een belangrijke beslissing worden genomen over de betekenis van elke rij in de feitentabel. Deze beslissing bepaalt bovendien hoe gedetailleerd de gegevens zullen zijn in het dimensionaal model. Kimball geeft de voorkeur aan het ontwikkelen van dimensionale modellen op basis van de meest atomaire gegevens die worden vastgelegd door het bedrijfsproces. Dit zijn de meest gedetailleerde feiten die men verzamelt en die niet verder kunnen worden onderverdeeld. Deze werkwijze is zinvol omwille van volgende redenen (Kimball en Ross 2002):

- Atomaire gegevens zijn erg dimensionaal. Hoe meer gedetailleerd een feitenmeting is, hoe meer gegevens men zeker weet. Alles waarvan men zeker is, kan worden vertaald in dimensies. In die zin zijn atomaire gegevens erg geschikt voor dimensionaal modelleren.
- Atomaire gegevens bieden een maximale analytische flexibiliteit omdat ze op zeer verschillende manieren kunnen worden verwerkt. Hierdoor zijn ze ideaal voor ad hoc zoekopdrachten van eindgebruikers.

Desalniettemin vormt het aggregeren van gegevens ter verbetering van de algemene prestaties, een belangrijk onderdeel van datawarehouses. Hoewel dit op het eerste zicht het gebruik van atomaire data lijkt tegen te spreken, worden enkele richtlijnen gegeven waardoor het meeste voordeel uit beide aspecten kan worden gehaald (Kimball en Ross 2002):

1. Door een goed inzicht in de informatiebehoeften van de eindgebruikers, kan worden vastgesteld welke gegevens frequent worden geaggregeerd.
2. Daarnaast dient men ook rekening te houden met de statistische verdeling van de gegevens. Indien bijvoorbeeld 50 producten worden gecategoriseerd in 10 merken, dan moet men gemiddeld slechts 5 rijen sommeren om een resultaat per merk te bekomen. In dit geval loont het niet de moeite om de gegevens vooraf te sommeren. Daarentegen is het gebruik van een aggregaat wel zinvol indien men telkens 100 rijen moet sommeren.

Stel dat een model wordt gebouwd voor het proces facturatie, dan vormt elke rij in de feitentabel een afzonderlijke rapportregel van de klantenfactuur.

Stap 3: De dimensies identificeren en conformeren

Dimensies zijn de drijvende kracht achter datamarts. Indien een dimensie in meerdere datamarts voorkomt, moeten dit precies dezelfde dimensies zijn, of moet de ene een

wiskundige deelverzameling van de andere zijn. Dit is de enige manier waarop twee datamarts één of meer dimensies in dezelfde toepassing kunnen delen. We spreken in dat geval van geconformeerde dimensies. Enkele goede voorbeelden van dimensies die absoluut tussen datamarts moeten worden geconformeerd, zijn de product- en klantdimensies in een onderneming.

Stap 4: De feiten kiezen

In de laatste stap worden de numerieke waarden geïdentificeerd die elke rij in de feitentabel zullen bezetten. Feiten worden bepaald door het beantwoorden van de vraag: 'Wat meten we?'. Een typisch voorbeeld is het aantal verkochte eenheden of het totale bedrag van verkochte producten.

Hoofdstuk 4

Datamining

Het doel van dit hoofdstuk is het introduceren van enkele datamining concepten. Zo wordt onder meer ingegaan op de betekenis, de evolutie en de voordelen van deze technologie. Verder worden ook de diverse stappen van het data mining proces overlopen en wordt een frequent toegepast datamining algoritme besproken.

4.1 Wat is datamining?

Bedrijven hebben gedurende de laatste decennia massale hoeveelheden gegevens verzameld. Het destilleren van bruikbare informatie uit de almaar groeiende stroom van ruwe data, vormt een zeer belangrijke uitdaging. Datamining is een proces voor het ontdekken van interessante informatie in grote hoeveelheden gegevens die zijn opgeslagen in databases, datawarehouses of andere opslagmogelijkheden. Ook wordt vaak de term *Knowledge Discovery* of kennisontdekking in Databases (KDD) gebruikt. Datamining analyseert de gegevens en ontdekt voor de gebruiker nuttige en relevante informatie. Met behulp van deze informatie kunnen trends en verbanden worden ontdekt waardoor bedrijven de mogelijkheid krijgen om proactief kennisgedreven beslissingen te nemen (Fernandez 2003). De basisvragen bij datamining zijn (Witteveen 2003):

1. Kunnen we nu een beter en efficiënter beleid voeren, door gebruik te maken van historische gegevens?
2. Kunnen we het toekomstige gedrag voorspellen als we de opgeslagen gegevens analyseren?
3. Welke onbekende informatie is te vinden in de gegevens die we in de loop der jaren reeds hebben verzameld?

De geautomatiseerde en prospectieve analyses die worden uitgevoerd door datamining gaan veel verder dan de analyses van historische gebeurtenissen die worden uitgevoerd door retrospectieve toepassingen, typisch voor beslissingsondersteunende systemen. Datamining applicaties kunnen

bedrijfsproblemen oplossen die voorheen te complex waren en teveel tijd in beslag namen. Door gebruik te maken van krachtige analytische technieken, stelt datamining organisaties in staat om kennis te ontginnen uit de groeiende hoeveelheid data en de daaruit resulterende beslissingsmodellen succesvol aan te wenden als hulpinstrument voor een verbeterde bedrijfsvoering (Thearling 1995). Zo blijkt datamining een competitief wapen te zijn (Groth 2000).

4.2 De evolutie van datamining

Datamining is het resultaat van een lang proces van onderzoek en productontwikkeling. De evolutie van datamining begon vanaf het moment dat bedrijfsgegevens voor de eerste maal werden opgeslagen met computers, en ging voort als gevolg van verbeteringen in de gegevenstoegang en meer recente technologieën die gebruikers toelaten om eenvoudig en snel doorheen gegevens te navigeren. Deze evolutie wordt weergegeven in tabel 4.1 (Thearling 1995).

Tabel 4.1: De evolutie van datamining

Fase	**Bedrijfsprobleem**	**Technologieën**	**Kenmerken**
Gegevensverzameling (1960)	"Hoeveel waren de totale opbrengsten gedurende de laatste vijf jaar?"	Computers, disketten	Retrospectieve, statische gegevens
Gegevenstoegang (1980)	"Hoeveel eenheden werden in maart verkocht in België?"	Relationele databases, Structured Query Language[1](SQL), ODBC[2]	Retrospectieve, dynamische gegevens op recordniveau
Datawarehousing en beslissingsondersteuning (1990)	"Hoeveel eenheden werden in maart verkocht in België, meer specifiek in Brussel?"	Online analytic processing (OLAP)[3], multidimensionale databases, datawarehouses	Retrospectieve, dynamische gegevens op meerdere niveau's
Datamining (nu)	"Hoeveel eenheden zullen volgende maand in Brussel worden verkocht en waarom?"	Geavanceerde algortimes, computers met meerdere processoren, grote databases	Prospectieve, proactieve informatie

[1]Gestandaardiseerde taal voor gegevensmanipulatie in relationele databasemanagementsoftware (Laudon en Laudon 2004)

[2]Open Database Connectivity, gestandaardiseerde toegangsmethode voor databases (Lycos 2005)

[3]Het analyseren van grote hoeveelheden data in multidimensionale databases voor het ontdekken van patronen, trends en uitzonderingen (O'Brien 1998)

Datamining kan vandaag de dag worden toegepast binnen de bedrijfsgemeenschap omwille van volgende technologieën:

1. ***De beschikbaarheid van grote databases en datawarehouses***
   Datamining dankt zijn naam aan het feit dat analisten op zoek gaan naar waardevolle informatie in gigantische databases. De laatste twee decennia, vond een explosieve groei plaats van de hoeveelheid gegevens die in een elektronische vorm worden opgeslagen. Daarnaast wordt ook een groei vastgesteld in het aantal datawarehouses (Fernandez 2003).

2. ***De prijsdaling van gegevensopslag en de toegenomen kracht van computers***
   De enorme daling van de prijs van gegevensopslag in slechts enkele jaren tijd, heeft een enorme invloed gehad op het verzamelen en opslaan van grote hoeveelheden gegevens. Ook is er een sterke daling in de kost van computerverwerking. Elke nieuwe generatie van chips, verhoogt de kracht van de centrale processor terwijl de prijs blijft dalen. Hoewel de kracht van de individuele processor enorm is toegenomen, boekt de architectuur met meerdere parallelle processoren de grootste vooruitgang (Two-Crows-Corporation 1999).

3. ***De vooruitgang in analytische methoden***
   Datamining haalt voordeel uit de evolutie van artificiële intelligentie (AI) en statistiek. Datamining is geen vervangmiddel voor de traditionele statistische methoden. Het is eerder een uitbreiding die gedeeltelijk is voortgekomen uit een grote verandering op het gebied van statistiek. De ontwikkeling van de meeste statistische technieken was tot voor kort gebaseerd op een elegante theorie en analytische methoden die goed werkten voor een bescheiden hoeveelheid data. Door de toegenomen kracht van computers en hun lage kostprijs, gekoppeld aan de behoefte om enorme hoeveelheden data te analyseren, werd de ontwikkeling van nieuwe technieken in de hand gewerkt. Een belangrijk punt is dat datamining deze en andere AI en statistische technieken toepast op algemene bedrijfsproblemen waarbij de productiviteit van de mensen die de modellen bouwen, wordt verhoogd (Two-Crows-Corporation 1999).

Datamining wordt steeds populairder door de substantiële bijdrage die het kan leveren, zowel voor het beheersen van kosten als voor het verhogen van opbrengsten. Datamining kan toegevoegde waarde leveren aan een breed spectrum van industriële sectoren. De telecommunicatiesector en de sector van kredietkaarten zijn de leiders in het toepassen van datamining om frauduleus gebruik van hun diensten vast te stellen. Maar ook verzekeringsmaatschappijen, farmaceutische bedrijven, ... maken vruchbaar gebruik van datamining (Two-Crows-Corporation 1999).

4.3 De voordelen van datamining

Er zijn twee sleutelfactoren voor het succes van datamining. Eerst en vooral moet men beschikken over een duidelijke formulering van het bedrijfsprobleem dat men wenst op te lossen. Een tweede voorwaarde is het gebruiken van geschikte gegevens. Gegeven een database van voldoende grootte en kwaliteit, dan kan datamining technologie nieuwe bedrijfsopportuniteiten creëren door in volgende mogelijkheden te voorzien (Thearling 1995):

- ***Geautomatiseerde ontdekking van voorheen onbekende patronen:*** Datamining toepassingen doorzoeken de databases grondig en kunnen daardoor voorheen verborgen patronen identificeren. Een voorbeeld van patroonherkenning is de analyse van de verkopen om op het eerste zicht ongerelateerde producten te vinden die vaak samen worden verkocht.

- ***Geautomatiseerde voorspelling van trends en gedrag:*** Datamining automatiseert het proces voor het zoeken naar voorspellende informatie in grote databases. Problemen die traditioneel een omslachtige handmatige analyse vereisten, kunnen nu vrijwel onmiddellijk op basis van de gegevens worden beantwoord. Een typisch voorbeeld van een voorspellend probleem is terug te vinden in marketing. Datamining maakt gebruik van gegevens van vroegere mailingen om zo de consumenten te identificeren die het meest waarschijnlijk zullen reageren op een toekomstige mailing.

Indien datamining applicaties worden geïmplementeerd op zeer krachtige computersystemen, dan zijn ze in staat om in slechts enkele minuten tijd gigantische databases te doorzoeken. Een snellere verwerking heeft tot gevolg dat gebruikers automatisch kunnen experimenteren met meerdere modellen om de complexe gegevens te leren begrijpen. De hoge snelheid maakt het praktisch voor de gebruiker om grote hoeveelheden data te analyseren, terwijl grote databases betere voorspellingen opleveren (Thearling 1995).

4.4 Het datamining proces

4.4.1 Algemeen

Het datamining proces bestaat uit volgende stappen (Fernandez 2003):

1. *Definitie van het bedrijfsprobleem*
   Eén van de voornaamste oorzaken voor het falen van datamining is het niet vooraf definiëren van de bedrijfsdoelstellingen. Op basis van goed geïdentificeerde bedrijfsproblemen kunnen de bedrijfsdoelen worden geformuleerd en kan een mogelijke oplossing worden aangereikt met behulp van datamining.

2. *Gegevensverwerking*

   De sleutel tot succesvol datamining is het gebruiken van geschikte gegevens. Het voorbereiden van data voor datamining is vaak het meest tijdrovende aspect van een datamining inspanning. Een typische databasestructuur voor datamining bevat observaties in rijen (bijvoorbeeld klanten, producten) en variabelen in kolommen (bijvoorbeeld demografische gegevens, verkoopsdata). Ook moet het meetniveau van elke variabele in de gegevensverzameling duidelijk worden gedefinieerd. Gegevens die numerieke waarden representeren, noemt men *continue variabelen*. Gegevens die behoren tot discrete klasses, noemt men *categorische variabelen* (bijvoorbeeld rood, blauw, groen). Categorische gegevens kunnen verder worden opgedeeld in ordinaal, met een zinvolle volgorde (bijvoorbeeld hoog/midden/laag), of nominaal, ongeordend (bijvoorbeeld postcodes) (Two-Crows-Corporation 1999). De stappen die onderdeel uitmaken van de verwerking van gegevens voor datamining zijn:

   - Opschonen van de gegevens: In dit stadium worden de gegevens opgeschoond zodanig dat onnodige gegevens die het zoeken zouden vertragen, worden verwijderd. Ook worden de data gecontroleerd op consistentie. Vaak zijn er in een database verschillende velden die dezelfde gegevens zouden moeten bevatten. Door typfouten of problemen bij het samenvoegen van verschillende databases, klopt dit niet altijd. Foute gegevens en extreme waarden moeten worden verwijderd uit de gegevensverzameling omdat ze de resultaten kunnen beïnvloeden waardoor de bruikbaarheid van de modellen wordt beperkt.
   - Gegevensintegratie: Hierbij wordt de vraag gesteld of de gegevens wel het antwoord bevatten op de vraag die gesteld is? Moeten externe gegevens aan de huidige gegevens worden toegevoegd? Moeten twee databases aan elkaar worden gekoppeld om de benodigde informatie te kunnen vinden? De gegevens uit de verschillende gegevensbronnen worden vervolgens gecombineerd in één enkele mining database en de verschillen in de gegevenswaarden worden geconsolideerd.
   - Transformatie van de variabelen: Soms is het uitdrukken van continue variabelen in meer gestandaardiseerde eenheden (bijvoorbeeld in log of vierkantswortel) noodzakelijk ter verbetering van de modellen. Imputatie van ontbrekende waarden is noodzakelijk indien belangrijke variabelen grote hoeveelheden ontbrekende waarden bevatten. Ook het identificeren van de doelvariabele en de inputvariabelen en het definiëren van de meeteenheid zijn voorname acties.
   - Opsplitsing van de database: Het nemen van een steekproef is aangeraden in zeer grote databases aangezien het de leertijd van het model aanzienlijk verkort. Ook het opdelen van de volledige dataset in training-, validatie- en testcategorieën is zeer belangrijk om het model beter te laten aansluiten en de modelresultaten te valideren.

Trends en patronen die worden waargenomen in de training dataset, kunnen worden veralgemeend naar de volledige dataset als de gebruikte training set de database voldoende representeert.

3. *Gegevensverkenning en beschrijvende analyse*
   Gegevensverkenning omvat een set van beschrijvende en grafische instrumenten die een visuele verkenning van de data mogelijk maken als voorwaarde voor een meer formele gegevensanalyse. Het vormt bovendien een integraal onderdeel van het bouwen van een model. Visualisatie van de gegevens laat gebruikers toe om de onderliggende structuur van de gegevens beter te begrijpen. Beschrijvende statistieken en verkennende grafieken zijn zeer handig bij het verkennen van continue en categorische variabelen.

4. *Datamining model bouwen: gericht leren versus ongericht leren*
   In deze stap worden verschillende datamining methoden toegepast met behulp van de computer. Men maakt een onderscheid tussen twee soorten datamining technieken namelijk gericht leren en ongericht leren. Gericht leren houdt in dat een model wordt gebouwd met het specifieke doel om op een optimale manier een doelveld in de historische database te voorspellen. Deze benadering maakt gebruik van training data die is samengesteld uit meerdere inputvariabelen en een doelvariabele. Vervolgens wordt een model gebouwd dat het doelveld tracht te voorspellen op basis van de inputvelden. Na de ontwikkeling van het model, kan het worden toegepast op nieuwe gelijkaardige gegevens, maar dan met een onbekende doelwaarde. Gerichte leermodellen omvatten classificatiemodellen en regressiemodellen. Classificatiemodellen maken gebruik van categorische doelvariabelen terwijl regressiemodellen continue en binaire doelvariabelen voorspellen. Ongericht leren daarentegen heeft geen enkel gedefinieerd doel om te voorspellen. Bij deze toepassing van datamining gaat men enkel op zoek naar patronen en verbanden om meer inzicht te bekomen in de gegevens.

5. *Modelvalidatie*
   Een belangrijke vereiste om de bruikbaarheid van het ontwikkelde model te bevestigen, is het valideren van modellen gebaseerd op training datasets met behulp van onafhankelijke validatie datasets. Modelvalidatie beoordeelt de kwaliteit van de afstemming van het model en beschermt tegen *overfitting*. Overfitting is het gevolg van een te scherpe afstemming van het model op de training dataset, waarin wellicht irrelevante onregelmatigheden aanwezig zijn. Hierdoor neemt het generalisatievermogen af. Modelvalidatie kan worden beschouwd als één van de voornaamste stappen in het proces.

6. *Interpretatie en besluitvorming*
   Tot slot moet de gevonden informatie zodanig worden opgeschreven dat diegenen die de

beslissingen moeten nemen, de resultaten ook begrijpen. Het nemen van juiste beslissingen is erg belangrijk voor een onderneming om succesvol te zijn. De patronen die worden geïdentificeerd door datamining modellen kunnen worden omgezet naar kennis, die vervolgens kan worden toegepast om bedrijfsbeslissingen te ondersteunen.

4.4.2 Problemen in het datamining proces

Vele datamining oplossingen die beschikbaar zijn op de markt kunnen moeilijk worden geïntegreerd, zijn niet schaalbaar en meestal beperkt tot één of twee modelleertechnieken. Hierdoor wordt steeds meer tijd besteed aan het verschaffen van toegang en het voorbereiden en manipuleren van gegevens uit verschillende gegevensbronnen. Zo wordt echter minder tijd besteed aan het modelleren van de data en het toepassen van de kennis uit deze modellen bij het oplossen van bedrijfsproblemen. Dit zal in de toekomst een nog grotere uitdaging vormen, doordat de hoeveelheid data en de complexiteit van bedrijfsproblemen blijven toenemen. Meestal zijn databases ontworpen voor andere doeleinden dan datamining waardoor eigenschappen en attributen die het leren zouden vereenvoudigen, niet aanwezig zijn. Ook worden heel wat problemen veroorzaakt doordat databases vaak zeer dynamisch, onvolledig en vervuild zijn (Fernandez 2003).

4.5 Datamining en datawarehousing

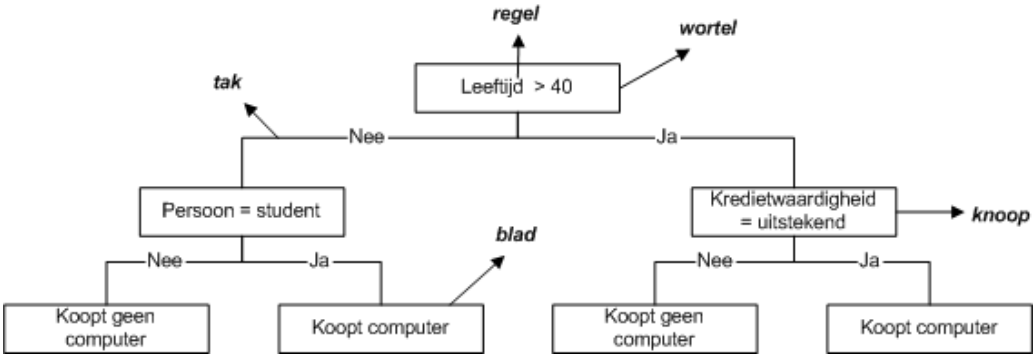
Een interessante trend in datamining, is de integratie van datawarehouse databases met datamining toepassingen. Wanneer men een datamining probleem aanpakt, is het eerst noodzakelijk om alle gegevens samen te brengen. In een realistische bedrijfstoepassing, moeten de gegevens van verscheidene afdelingen worden verzameld. Zo zal bijvoorbeeld een marketingstudie gegevens nodig hebben van de verkoopsafdeling, de boekhouding en de klantendienst. Het integreren van gegevens uit verschillende bronnen brengt meestal een aantal praktische problemen met zich mee. Verschillende afdelingen maken vaak gebruik van verschillende opslagtechnieken, verschillende conventies, verschillende tijdsperioden, verschillende graden van gegevensaggregatie en verschillende primaire sleutels. De gegevens moeten dus worden verzameld, geïntegreerd en opgeschoond. Het idee van een gegevenswijde database-integratie staat bekend als datawarehousing. Datawarehouses voorzien een gecentraliseerde toegang tot de bedrijfsgegevens die afkomstig zijn van verschillende afdelingen. Ze bevatten historische gegevens waarop bedrijfsbeslissingen kunnen worden gebaseerd. De verschuiving richting datawarehousing is een erkenning van het feit dat de gefragmenteerde gegevens die een organisatie gebruikt om haar dagelijkse operaties op afdelingsniveau te ondersteunen, een imense strategische waarde kunnen opleveren wanneer ze worden samengebracht (Witten en Frank 2000). Daarom zijn datamining applicaties die gebruik maken van de gegevenspreparatie- en gegevensintegratiecapaciteiten van datawarehouses beter in staat om goede resultaten te bekomen.

Eén van de sleutelfactoren voor succesvol datamining is de mogelijkheid om toegang te krijgen tot accurate, volledige en geïntegreerde gegevens. Dit geldt ook voor datawarehousing. Een datawarehouse is niet alleen een middel voor gegevensintegratie, maar alle datawarehouse-oplossingen starten met en zijn afhankelijk van de kwaliteit en effectiviteit van hun gegevensbronnen (Berson *et al.* 1999).

4.6 Beslissingsbomen

Datamining maakt gebruik van verschillende recent ontwikkelde methoden. In deze paragraaf lichten we één van de meest gebruikte methoden toe. Voor andere methoden en overwegingen die in het datamining proces een rol spelen, wordt verwezen naar de literatuur.

Een beslissingsboom is een grafische voorstelling van een gegevensverzameling door middel van een boomstructuur. Een beslissingsboom is opgebouwd uit knopen, takken en bladeren (figuur 4.1). Een boom start vanuit de wortelknoop, die alle observaties uit de gegevensverzameling bevat, en groeit verder naar beneden door de observaties op elk niveau te splitsen in nieuwe knopen. Elk van deze knopen bevat een deel van de oorspronkelijke waarnemingen. Alle knopen zijn met elkaar verbonden via takken. Knopen die zich aan het einde van een tak bevinden, worden bladeren genoemd (Negnevitsky 2001).



Figuur 4.1: Voorbeeld van een beslissingsboom: Koopt computer?

Een beslissingsboom maakt gebruik van inputvariabelen om waarnemingen te splitsen in verschillende groepen die discrimineren naar de te verklaren variabele (doelvariabele). Het gegevensbestand wordt hierdoor opgedeeld in elkaar uitsluitende segmenten op basis van de doelvariabele. Deze segmentatie komt tot stand door na te gaan welke variabele en welke samenvoeging van waarden binnen deze variabele het hoogst mogelijk onderscheidend vermogen heeft ten aanzien van de doelvariabele. Dit proces wordt iteratief toegepast totdat er geen onderverdeling

meer mogelijk is die voldoende onderscheidend vermogen bezit (Blomme 2004).

Een voordeel van beslissingsbomen is dat de meest belangrijke variabelen en de combinaties (en interacties) ertussen voor de voorspelling van de doelvariabele worden gedecteerd. Op basis van beslissingsbomen worden regels opgesteld die aangeven aan welke voorwaarden een waarneming dient te voldoen om in een bepaald segment terecht te komen. Het segment geeft de voorspellende waarde aan voor de observaties die daarbinnen vallen (Blomme 2004).

Beslissingsbomen worden voor diverse doeleinden aangewend (Berson en Smith 1997):

1. Een beslissingsboom kan worden gebruikt voor de exploratie van de gegevensverzameling en het bedrijfsmatige probleem. Dit wordt gedaan door naar de voorspellende variabelen en waarden te kijken die voor elke vertakking van de boom worden gekozen. Deze voorspellende variabelen bieden vaak een bruikbaar inzicht of een voorstel voor vragen die moeten worden beantwoord.

2. Een andere toepassing waarvoor beslissingsbomen worden gebruikt, is het vooraf bewerken van gegevens voor andere voorspellende algoritmen. Beslissingsbomen kunnen tijdens de eerste stap van een datamining sessie een deelverzameling van mogelijk voorspellende variabelen creëren.

3. Tenslotte worden beslissingsbomen steeds vaker gebruikt als voorspellende modellen.

De visualisering van de resultaten en de hoge interpreteerbaarheid van beslissingsbomen hebben ertoe geleid dat deze techniek tegenwoordig veel wordt toegepast. Zowel voor classificatie als voor regressie is de techniek van beslissingsbomen een aangewezen analysemiddel (Blomme 2004).

Hoofdstuk 5

Predictive Model Markup Language

In dit hoofdstuk wordt een op XML (eXtensible Markup Language) gebaseerde standaard geïntroduceerd voor het beschrijven van datamining modellen, namelijk Predictive Model Markup Language of PMML. Het eerste deel van dit hoofdstuk is een algemene inleiding voor de lezer die onbekend is met het begrip XML. In het daarop volgende deel worden alle aspecten van PMML belicht.

5.1 eXtensible Markup Language (XML)

5.1.1 Wat is XML?

XML is een door het World Wide Web Consortium[1] (W3C) aanbevolen standaard voor het beschrijven van gestructureerde data op het internet. Als dusdanig stelt XML communicerende partijen in staat om gegevens uit te wisselen via een flexibel formaat, dat onafhankelijk is van applicaties en platformen (Van Dijk 1999).

XML is een vereenvoudigde versie van de Standard Generalized Markup Language (SGML) die in 1974 werd ontwikkeld door IBM als onderdeel van een project voor het uitwisselen van documenten. Van deze standaard is later ook HyperText Markup Language (HTML) afgeleid: De taal die wordt gebruikt op het World Wide Web (WWW) voor het opmaken van de lay-out van documenten (Benz en Durant 2003). Ook XML is in eerste instantie bedoeld voor gebruik op het internet. SGML was hiervoor minder geschikt en werd daarom vereenvoudigd tot een taal die gemakkelijk te gebruiken en te implementeren is. Een ruwe schatting is dat XML 80 procent van de functionaliteit van SGML bereikt met 20 procent van de complexiteit (Van Dijk 1999).

[1] Een consortium van commerciële en educatieve instellingen dat toezicht houdt op het onderzoek en de standaarden promoot in alle gebieden die zijn gerelateerd aan het World Wide Web (Ondersteuning van Microsoft Office 2005).

XML tracht tegemoet te komen aan de tekortkomingen van HTML, dat oorspronkelijk was bedoeld als opmaaktaal voor statische documenten. Aangezien HTML gericht is op de presentatie van gegevens, is het moeilijk om iets anders met deze gegevens te doen dan ze te bekijken. Er bestaat daarom een behoefte aan het scheiden van data en presentatie: HTML is geschikt voor het visualiseren van data, terwijl de data zelf op een andere manier wordt beschreven. Speciaal voor dit doel werd XML ontwikkeld (Van Dijk 1999).

XML is een markeertaal. Dit wil zeggen dat met behulp van XML een document kan worden voorzien van een opmaak die het structureren en beschrijven van gegevens in dat document, ten bate van een bepaalde toepassing, eenvoudiger maakt. De kracht van XML ligt juist in het feit dat het een dynamische opmaaktaal is. De toepassing ligt niet van te voren vast. Een XML document kan als het ware worden geparametriseerd door een documenttype, dat aangeeft wat de precieze toepassing van het document is. De term eXtensible Markup Language is wat dat betreft enigszins verwarrend: XML is niet een taal voor de opmaak van een specifiek type document, maar een standaard syntax voor een oneindige verzameling van verschillende typen documenten. XML wordt daarom ook wel een metataal genoemd (Van Dijk 1999).

De waarde van XML is gebaseerd op drie belangrijke eigenschappen (Benz en Durant 2003):

1. **Uitbreidbaarheid**
   XML staat toe dat iedereen zijn eigen structuur definieert. Daarnaast is het mogelijk bestaande structuren uit te breiden door het toevoegen van een nieuwe structuur.

2. **Structureerbaarheid**
   Met XML is het mogelijk om willekeurige complexe, hiërarchische data te representeren in een flexibel formaat. De structuur, volgens welke de data worden opgeslagen, kan bovendien worden vastgelegd in het document zelf. De structuur is per definitie ondubbelzinnig: Het is niet mogelijk de structuur op twee verschillende manieren te interpreteren.

3. **Valideerbaarheid**
   XML documenten kunnen eenvoudig worden gecontroloreerd op juistheid, dit wil zeggen: Er kan worden gecontroleerd of de structuur van de gegevens overeenkomt met de definitie van die structuur. Hiervoor zijn twee standaarden voorzien namelijk de oorspronkelijke Data Type Definition (DTD) en de meer recente XML Schema standaard. Deze begrippen zullen later in dit hoofdstuk aan bod komen.

Bovenstaande eigenschappen maken XML tot een krachtige standaard voor zeer uiteenlopende doeleinden. Voor een beter begrip van XML volgt eerst een nadere beschrijving van de XML syntax.

5.1.2 De structuur van een XML document

Een belangrijke eigenschap van XML documenten is dat ze zowel leesbaar zijn voor machines als voor mensen; ze worden gepresenteerd als normale tekst. XML documenten kunnen onder meer elementen, attributen en naamruimten bevatten.

Elementen

Een XML document bestaat uit gestructureerde data. Deze data zijn ondergebracht in zogeheten elementen. Elk element heeft een type en een inhoud. De inhoud kan ofwel leeg zijn ofwel bestaan uit tekst of uit andere elementen (Benz en Durant 2003). Een eenvoudig voorbeeld van een element is:

<voornaam>Lene</voornaam>

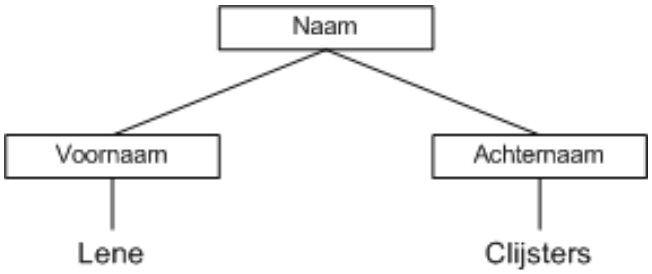
Dit element heeft als type *voornaam*, en als inhoud de tekst *Lene*. De inhoud van een element wordt omgeven door tags, die aangeven waar een element begint (de openingstag) en waar het eindigt (de sluitingstag). Een openingstag wordt altijd weergegeven door de naam van het element te omgeven door < en > tekens, terwijl een sluitingstag bovendien nog een / na de < bevat. Elementen kunnen echter ook uit andere elementen bestaan, bijvoorbeeld:

<naam>
<voornaam>Lene</voornaam>
<achternaam>Clijsters</achternaam>
</naam>

Dit XML document bestaat uit drie elementen, waarbij het element van het type *naam* is samengesteld uit de types *voornaam* en *achternaam*. Elementen die geen attributen of tekst bevatten worden als volgt voorgesteld (Benz en Durant 2003):

<naam/>

Een XML document heeft een hiërarchische structuur die grafisch kan worden weergegeven door middel van een boomstructuur (figuur 5.1).



Figuur 5.1: Boomstructuur

Attributen

Elementen kunnen verder ook attributen of kenmerken hebben. Hiermee kan extra informatie worden opgeslagen bij een element, die niet expliciet in de uitvoer voor de gebruiker wordt getoond (Benz en Durant 2003), bijvoorbeeld:

<naam geslacht="vrouw">Lene Clijsters</naam>

Een attribuut is een combinatie van een naam en een waarde, gescheiden door een gelijkheidsteken, die deel uitmaakt van een element. Alle attribuutwaarden zijn tekenreeksen in tekstvorm en moeten bijgevolg tussen aanhalingstekens staan.

Naamruimten

XML naamruimten worden gebruikt om elementen en attributen binnen een XML document een unieke aanduiding te geven zodat naamconflicten worden voorkomen. Een XML naamruimte wordt gedeclareerd als een Uniform Resource Locator (URL) en bevindt zich gewoonlijk in de begincode van het hoofdelement van een XML document. Een URL is een adres dat een protocol bevat en een locatie van een object, document, webpagina of andere bestemming op het internet of op een intranet. Als aan een naamruimte een URL wordt gekoppeld, kunnen elementen altijd van elkaar worden onderscheiden ook al hebben ze dezelfde naam (W3 Schools 2005).

De aanbeveling van het W3C over het gebruik van naamruimten in XML heeft het attribuut *xmlns* opgeleverd, waarmee een unieke naamruimte voor een XML document kan worden gedefinieerd. Een voorbeeld van een declaratie van een naamruimte is:

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

Een naamruimte bestaat uit drie delen (W3 Schools 2005):

- **xmlns-sleutelwoord:** Het sleutelwoord dat door het W3C is gedefinieerd voor het aanduiden van XML naamruimten. Dit gedeelte wordt als eerste gedefinieerd en wordt van het voorvoegsel van de naamruimte gescheiden door een dubbel punt.
- **Voorvoegsel:** De afkorting van de naam van de naamruimte die wordt gebruikt om alle elementen en attributen die bij de naamruimte horen, te declareren. Als voorvoegsel kan elke naam worden gebruikt, behalve xmlns of xml, aangezien dit gereserveerde woorden zijn.
- **Definitie:** De URL die de unieke aanduiding van de naamruimte vormt.

Nadat een naamruimte is gedeclareerd voor een XML document, wordt in de declaratie van elk element of attribuut dat tot die naamuimte behoort, het voorvoegsel van de naamruimte gebruikt. Aan de hand van onderstaand eenvoudig voorbeeld, illustreren we het gebruik van naamruimten.

```
<departement>
   <naam>AFD1</naam>
   <adr:adres xmlns:adr="http://www.tu-darmstadt.de/ito/adressen">
      <adr:straat>Stukkenstraat 24</adr:straat>
      <adr:gemeente>Bree</adr:gemeente>
   </adr:adres>
   <serv:server xmlns:serv="http://www.tu-darmstadt.de/ito/servers">
      <serv:naam>OnzeWebServer</serv:naam>
      <serv:adres>123.45.67.8</serv:adres>
   </serv:server>
</departement>
```

In dit voorbeeld wordt getoond hoe het elementtype *adres* op twee verschillende manieren wordt gebruikt. In de derde regel wordt een naamruimte gedeclareerd voor het elementtype *adres* met als voorvoegsel *adr*. Enkele regels verder zien we het element *adres* opnieuw verschijnen, maar met een ander voorvoegsel en dus ook een andere naamruimte. Door toepassing van naamruimtes kan een unieke naam worden toegekend aan het elementtype *adres* dat in meerdere betekenissen kan voorkomen. Het elementtype *adr:adres* beschrijft een specifieke locatie terwijl *serv:adres* het adres van een server weergeeft.

5.1.3 Juist gevormde XML documenten

Het World Wide Web Consortium heeft een aantal strikte vereisten gedefinieerd voor de vorm van een XML document (W3C Recommendation 2005). XML documenten die voldoen aan

de vormgevingsregels van W3C, worden omschreven als *well-formed* of juist gevormde XML documenten. De vormgevingsregels kunnen worden herleid tot zes basisregels:

1. De eerste regel van een XML document bevat de XML declaratie:
   <?xml version="1.0">.

2. Elk element moet bestaan uit een openingstag en een sluitingstag. Een uitzondering op deze regel zijn de lege elementen, waarbij de openingstag en de sluitingstag worden gecombineerd.

3. Er moet juist één *root* of hoofdelement bestaan die alle andere elementen omvat.

4. Alle tags moeten juist genest zijn.

5. Attributen moeten tussen aanhalingstekens staan.

6. De gereserveerde tekens in XML (bijvoorbeeld: >, < en &) moeten correct worden gebruikt.

5.1.4 XML validatie

Door verschillende elementen met elkaar te combineren kunnen willekeurige boomstructuren worden gemaakt. Elk element is in principe te combineren met elk ander element. Vaak is deze vrijheid om combinaties van elementen te maken niet gewenst. In veel gevallen zal slechts een beperkt aantal typen elementen worden gebruikt, en vaak moeten deze elementen in een bepaalde volgorde staan. Om dergelijke restricties op de structuur van XML documenten te kunnen leggen, kan men gebruik maken van Document Type Definition (DTD) of XML schema's. XML documenten worden vergeleken met de regels die gespecificeerd zijn in een DTD of schema. Een juist gevormd XML document dat voldoet aan alle vereisten wordt een geldig XML document genoemd (Benz en Durant 2003).

5.1.4.1 Validatie met behulp van Document Type Definition (DTD)

Document Type Definition (DTD) is de oorspronkelijke methode voor het valideren van XML documenten. Een DTD is een verzameling van regels waaraan het XML document moet voldoen. Men kan een DTD beschouwen als een blauwdruk. Het geeft een beschrijving van de mogelijke structuur van een XML document samen met de tags. DTD's worden geschreven in een aparte syntax en zijn bijgevolg geen juist gevormde XML documenten. Een DTD legt restricties op voor elementnamen, attribuutnamen en waarden, de volgorde van de elementen en de nesting van de elementen (Benz en Durant 2003). Een eenvoudig voorbeeld van een DTD is:

<! ELEMENT naam ((voornaam, achternaam) — (achternaam, voornaam))>
<! ELEMENT voornaam (#PCDATA)>
<! ELEMENT achternaam (#PCDATA)>

Deze DTD geeft aan dat het elementtype *naam* uit twee subelementen bestaat namelijk *voornaam* en *achternaam* die in willekeurige volgorde mogen staan. De elementen *voornaam* en *achternaam* mogen enkel een reeks van karakters bevatten en hebben geen subelementen.

5.1.4.2 Validatie op basis van een XML schema

XML schema's zijn de opvolgers van DTD's. Een XML schema wordt gebruikt om de structuur en het type gegevens dat een XML document kan bevatten, te definiëren. In een XML schema worden de elementen, attributen en gegevenstypen gespecificeerd die in een XML document kunnen worden gebruikt. Daarnaast wordt in het schema de structuur aangegeven die het XML document moet volgen om geldig te zijn voor dat bepaald XML schema. Het XML schema kan bovendien worden gebruikt om de gegevens in de elementen en attributen te verifiëren. Hierna volgt een lijst van enkele typen gegevensverificatie die door een XML schema kunnen worden uitgevoerd (eXtensible Markup Language 2005):

- **Verificatie van gegevenstypen:** In een XML schema kan worden bepaald welke gegevenstypen een element of attribuut kan bevatten.
- **Verificatie op basis van beperkingen:** In een XML schema kunnen beperkingen worden gesteld aan de ruimte voor waarden van gegevenstypen.
- **Verificatie op basis van aantal:** In een XML schema kan het aantal toegestane exemplaren worden bepaald.
- **Verificatie door keuzebeperking:** In een XML schema kunnen de toegestande waarden worden beperkt tot de waarden die in een lijst met waarden voorkomen.
- **Verificatie op basis van volgorde:** In een XML schema kan de volgorde worden bepaald waarin elementen mogen worden gebruikt.
- **Standaardwaarden:** In een XML schema zijn dit de waarden die worden gebruikt als er geen andere waarde is opgegeven.

In tegenstelling tot DTD's, volgen XML schema's wel de regels van juist gevormde XML documenten. Als gevolg van het XML formaat en de gedetailleerde controles, zijn schema's over het algemeen veel uitgebreider dan de XML documenten die ze beschrijven. Daar tegenover staat dat schema's vaak gemakkelijker te begrijpen zijn voor ontwikkelaars (W3 Schools 2005).

Schemabestanden hebben een hoofdelement met de naam *schema*. Alle definities van elementen, attributen en gegevenstypen zijn in het schema genest. Er wordt een onderscheid gemaakt tussen simpele elementen, attributen en complexe elementen. Een **simpel element** is een XML element dat enkel en alleen tekst bevat. Het kan dus geen andere elementen of attributen omvatten, maar wel verschillende soorten tekst. De declaratie van een simpel element definieert de naam en het gegevenstype van het element, bijvoorbeeld (W3 Schools 2005):

<xs:element name="achternaam" type="xs:string" />
<xs:element name="leeftijd" type="xs:integer" />
<xs:element name="geboortedatum" type="xs:date" />

Dit schema geeft aan dat het element *achternaam* in een reeks van karakters moet worden uitgedrukt, dat het element *leeftijd* een geheel getal is en dat het element *geboortedatum* een geldige datum moet zijn.

Simpele elementen hebben geen attributen. Als een element attributen heeft, wordt het beschouwd als een complex element. Het **attribuut** zelf echter, wordt gedeclareerd als een simpel type. Een attribuut wordt als volgt gedefinieerd (W3 Schools 2005):

<xs:attribute name="taal" type="xs:string" />

Een **complex element** is een XML element dat andere elementen en/of attributen bevat. Onderstaand voorbeeld illustreert hoe een complex element wordt gedeclareerd (W3 Schools 2005):

```
<xs:element name="student" type="persooninfo" />
<xs:complexType name="persooninfo">
   <xs:sequence>
      <xs:element name="voornaam" type="xs:string" />
      <xs:element name="achternaam" type="xs:string" />
   </xs:sequence>
   <xs:attribute name="taal" type="xs:string" />
</xs:complexType>
```

Het element *student* heeft een gegevenstype dat aangeeft tot welk complexe type het element behoort, namelijk *persooninfo*. Vervolgens wordt het complexe type *persooninfo* gedeclareerd. De indicator *<sequence>* duidt aan dat de simpele elementen *voornaam* en *achternaam* in de volgorde moeten voorkomen zoals aangegeven in het schema. Verder heeft dit complexe type ook nog een taalattribuut.

5.2 Predictive Model Markup Language (PMML)

Deze paragraaf is grotendeels gebaseerd op de PMML 2.1 specificatie die beschikbaar is op de website van de Data Mining Group (Data Mining Group 2005) .

5.2.1 Wat is PMML?

De Predictive Model Markup Language (PMML) is een op XML gebaseerde markeertaal voor het definiëren en uitwisselen van datamining modellen. Het beschrijft de input van het model, de transformaties die nodig zijn bij het voorbereiden van de gegevens voor datamining en de parameters van het model. Met behulp van PMML is het mogelijk om eenvoudig en snel datamining modellen uit te wisselen tussen toepassingen van diverse softwareleveranciers (Buchner *et al.* 1998). Daardoor wordt het bijvoorbeeld mogelijk om een model te ontwikkelen met behulp van een bepaalde toepassing, en nadien een andere toepassing te gebruiken voor het visualiseren, analyseren of evalueren van dit model.

PMML werd ontwikkeld door de Data Mining Group (DMG). Dit is een onafhankelijke werkgroep die wordt geleid door verscheidene grote softwareproducenten en die zich richt op de ontwikkeling van datamining standaarden. Op dit ogenblik zijn de voornaamste leden: IBM, KXEN, Magnify, Microsoft, MicroStrategy, National Center for Data Mining, Oracle, Prudential Systems Software, Salford Systems, SAS, SPSS en StatSoft. De eerste versie van PMML (1.0) verscheen in augustus 1999. Nadien volgde versie 1.2 (2000), versie 2.0 (2001) en versie 2.1 (2003). Inmiddels werd versie 3.0 uitgebracht (oktober 2004). De eerste versies van PMML bestonden uit sets van DTD's. Vanaf versie 2.1 werd overgeschakeld op XML schema's.

PMML biedt onder meer volgende voordelen (Wettschereck en Muller 2001):

1. Het maakt de gegevens die worden geëxtraheerd, onafhankelijkheid van de toepassing, de implementatie, het hardwareplatform en het besturingssysteem.
2. Het vereenvoudigt het gebruik van datamining modellen door andere toepassingen of personen. Bijvoorbeeld: consultants kunnen handelen als ontwikkelaars van modellen. Hun klanten kunnen deze modellen vervolgens importeren in hun eigen toepassingen.
3. Het concentreert zich niet op het creëren van modellen of het implementeren van algoritmes.

5.2.2 Algemene structuur van een PMML document

PMML maakt gebruik van XML voor de voorstelling van datamining modellen. De structuur van de modellen wordt beschreven aan de hand van XML schema's. Een PMML document

heeft meestal volgende opbouw:

<?xml version=" 1.0" >
<PMML version=" 2.1"
xmlns="http://www.dmg.org/PMML-2_1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<Header>...</Header>
<DataDictionary>...</DataDictionary>
<TransformationDictionary>...</TransformationDataDictionary>
...MODEL...
</PMML>

Het hoofdelement van een PMML document moet van het type PMML zijn. Daarbij moet ook de versie als attribuut van het hoofdelement worden opgegeven. Een geldig PMML 2.1 document moet een XML document zijn dat voldoet aan het XML schema dat zich bevindt op http://www.w3.org/2001/XMLSchema-instance. Daarnaast moet het document beantwoorden aan de beperkingen die worden weergegeven in de modelspecificaties (http://www.dmg.org/-PMML-2_1). De voornaamste componenten waaruit een PMML document is opgebouwd, zijn:

- Hoofding (Header)
- Gegevenswoordenboek (Data Dictionary)
- Transformatiewoordenboek (Transformation Dictionary)
- Model
  - Mining schema
  - Locale transformaties (Local Transformations)
  - Model parameters

Deze componenten worden elk individueel behandeld in de volgende secties.

5.2.3 Componenten van een PMML document

5.2.3.1 Hoofding (Header)

De header of hoofding wordt aangegeven met behulp van het element *<header>* en bevat algemene informatie over de modellen die worden beschreven in het PMML document. De hoofding heeft één verplicht attribuut, namelijk het copyright, en een optioneel attribuut dat eventueel extra informatie bevat over het gebruik van het model in andere toepassingen. Verder kan de hoofding volgende elementen omvatten:

- *<Application>*
  Dit element beschrijft de toepassing waarmee het model werd ontwikkeld en heeft als attributen de naam (verplicht) en de versie van de applicatie. Hoewel PMML modellen zodanig worden gecreëerd dat ze uitwisselbaar zijn, is het mogelijk dat verschillende mechanismen andere modellen creëren op basis van dezelfde gegevensverzameling. Daarom is het belangrijk voor de gebruiker om te weten welke applicatie het model voortbracht.

- *<Annotation>*
  Met behulp van dit element kan extra informatie of kunnen opmerkingen worden doorgegeven in geval van een wijziging van het document.

- *<Timestamp>*
  Dit element geeft het tijdstip aan waarop het model werd gecreëerd.

5.2.3.2 Gegevenswoordenboek (Data Dictionary)

Het data dictionary of gegevenswoordenboek bevat definities van de variabelen uit de gegevensbron. Het geeft voor elke variabele zowel het gegevenstype als het bereik van de waarden weer. Het gegevenswoordenboek is gemeenschappelijk voor alle modellen in het PMML document en wordt gedeclareerd aan de hand van het element *<DataDictionary>*. Dit element heeft slechts één attribuut namelijk *numberOfFields* of het aantal variabelen dat in het gegevenswoordenboek wordt gedefinieerd. Deze waarde dient als een extra controle op de consistentie van het document. Het declareren van variabelen gebeurt door middel van het element *<DataField>*. Dit element definieert niet alleen een specifieke variabele, maar geeft ook de waarden aan die als geldig worden beschouwd. De voornaamste attributen van dit element zijn:

- *name:* Naam van de variabele, deze naam moet uniek zijn binnen het gegevenswoordenboek (verplicht).

- *displayName:* Naam die door toepassingen kan worden gebruikt om naar de variabele te verwijzen. Binnen het PMML document is echter enkel de naam van de variabele van belang.

- *opType:* Het type van de variabele op basis van de bewerkingen die gedefinieerd zijn voor de waarden van de variabele. Dit verplicht attribuut heeft als mogelijke waarden: categorisch (hebben de operator =), ordinaal (hebben naast de operator = ook < en >) of continu (hebben niet alleen =,< en > maar ook rekenkundige operatoren).

- *dataType:* Het gegevenstype van de waarden van de variabele.

De inhoud van een *<DataField>* element definieert een verzameling van waarden die geldig zijn. Datamining modellen onderscheiden drie waarden:

- Ontbrekende waarden
- Ongeldige waarden (liggen niet binnen het bereik van het gegeven interval)
- Geldige waarden

Voor het definiëren van het bereik van de waarden van een specifieke variabele in het gegevenswoordenboek, wordt één van volgende elementen gebruikt:

- *<Value>*
  Dit element definieert een waarde en heeft drie attributen. Het eerste attribuut, namelijk *value*, is verplicht en bevat een specifieke waarde. Het attribuut *displayValue* is optioneel en laat toe een verduidelijking te geven van een cryptische waarde. Tenslotte geeft het attribuut *property* aan of het om een geldige, ongeldige of ontbrekende waarde gaat.
- *<Interval>*
  Dit element definieert een interval van numerieke waarden. De grenzen van het interval worden aangeduid met behulp van het verplicht attribuut *closure*. Dit attribuut kan vier waarden aannemen namelijk openGesloten, openOpen, geslotenOpen of geslotenGesloten. De linker en rechter grens van het interval worden respectievelijk aangegeven door de attributen *leftMargin* en *rightMargin*. Deze attributen zijn optioneel, maar minstens één van beiden moet worden ingevuld. Een waarde die niet is opgegeven, wordt beschouwd als oneindig.

5.2.3.3 Transformatiewoordenboek (Transformation Dictionary)

Datamining modellen maken soms gebruik van eenvoudige functies om gebruikersgegevens te converteren naar waarden waarmee modellen gemakkelijker kunnen rekenen. PMML definieert hiervoor volgende transformaties:

- **Normalisatie** zet waarden om in getallen, de input kan zowel continu als discreet zijn.
- **Discretisatie** zet continue waarden om in discrete waarden.
- **Waarde overeenstemming (*Value mapping*)** zet discrete waarden om naar andere discrete waarden.
- **Aggregatie** sommeert of verzamelt gegevensgroepen, bijvoorbeeld door het berekenen van het gemiddelde.

PMML omschrijft niet alle mogelijke transformaties voor het voorbereiden van gegevens voor datamining, aangezien deze te talrijk zijn. In plaats daarvan bevat PMML enkel die transformaties die automatisch worden gecreëerd door een datamining systeem.

Getransformeerde of afgeleide variabelen worden gedefinieerd met behulp van *<DerivedField>* elementen. De *<DerivedField>* elementen die ondergeschikt zijn aan het element *<TransformationDictionary>*, kunnen door meerdere modellen in het PMML document worden gebruikt. Daarnaast kan elk model ook zijn eigen afgeleide variabelen definiëren door middel van het element *<LocalTransformations>*. Ook zijn er modellen die onmiddellijk gebruik maken van *<DerivedField>* elementen in hun definitie.

Het element *<DerivedField>* omvat een transformatie-uitdrukking die aangeeft via welke methode de nieuwe waarden worden berekend. Daarnaast heeft een afgeleide variabele een *name* en *displayName* attribuut. Afgeleide elementen die opgenomen zijn in een *<transformationDictionary>* of in een *<LocalTransformations>* element, moeten een unieke naam hebben. Voor afgeleide variabelen die in een model zijn opgenomen, is het naamattribuut optioneel.

5.2.3.4 Mining Schema

Elk datamining model heeft een mining schema, dat een overzicht biedt van de variabelen die in het model worden gebruikt. Dit is een deelverzameling van de variabelen die gedefinieerd zijn in het gegevenswoordenboek. De namen van de variabelen in het mining schema moeten uniek zijn opdat het PMML document geldig is. In tegenstelling tot het mining schema dat informatie bevat voor een specifiek model, bevat het gegevenswoordenboek gegevensdefinities die algemeen voor elk model gelden. De hoofddoelstelling van het mining schema is het opsommen van de variabelen die de gebruiker moet voorzien opdat het model kan worden toegepast.

Het begin en einde van het mining schema worden aangegeven door middel van *<MiningSchema>*. Dit element heeft geen attributen en kan enkel het element *<MiningField>* bevatten. Het element *<MiningField>* heeft volgende kenmerken:

- *name:* De naam van de variabele die verwijst naar een variabele in het gegevenswoordenboek (verplicht).
- *usageType:* De wijze waarop de variabele zal worden gebruikt in het model. Mogelijke waarden zijn: actief (inputvariabale), voorspellend (doelvariabele), supplementair (additionele beschrijvende variabele) en gegroepeerd (variabele op basis waarvan een groepering wordt gemaakt).
- *outliers:* Geeft aan op welke wijze wordt omgegaan met uitzonderlijke waarden. Mogelijke manieren zijn:
  - asIs: Uitschieters worden beschouwd als normale waarden.
  - asMissingValues: Uitschieters worden beschouwd als ontbrekende waarden.

  - asExtremeValues: Uitschieters worden vervangen door een vooraf bepaalde hoge of lage waarde.

- *lowValue/highValue:* Wordt in conjunctie gebruikt met de methode *asExtremeValues*. Indien een waarde kleiner is dan de lage waarde (*lowValue*) of groter is dan de hoge waarde (*highValue*) dan wordt de lage waarde (*lowValue*) respectievelijk de hoge waarde (*highValue*) gebruikt in plaats van de oorspronkelijke waarde.

- *missingValueReplacement:* Indien dit attribuut gespecificeerd is, wordt elke ontbrekende waarde automatisch vervangen door de opgegeven waarde.

- *missingValueTreatment:* Geeft aan welke methode werd toegepast om om te gaan met ontbrekende waarden. Dit attribuut is echter louter informatief.

Ontbrekende waarden komen op diverse plaatsen terug in een PMML document:

1. De externe voorstelling van ontbrekende waarden wordt niet direct gedefinieerd door PMML, maar is afhankelijk van de toepassing die het PMML document verwerkt.

2. In het gegevenswoordenboek kan een optionele lijst van waarden worden opgegeven die aangeven dat een waarde ontbreekt (bijvoorbeeld: NA of -).

3. Het mining schema van een model definieert een optionele vervangingswaarde.

4. Voor elk modeltype in PMML bestaat een methode voor het opnemen van ontbrekende waarden in de berekening van de resultaten.

5.2.3.5 Locale transformaties (Local Transformations)

Locale transformaties zijn vergelijkbaar met het transformatiewoordenboek, maar gelden enkel voor een specifiek model (zie paragraaf 5.2.3.3).

5.2.3.6 Model parameters

Elk datamining model wordt gedefinieerd aan de hand van een aantal parameters. Een PMML document kan zowel meerdere als geen modellen bevatten. In het laatste geval zal het document worden gebruikt voor het leveren van initiële metagegevens. PMML 2.1 ondersteunt volgende datamining modellen: associatieregels, clustering, regressies, naive bayes, neurale netwerken, reeksen en beslissingsbomen. We beperken ons hier tot het bespreken van beslissingsbomen.

Beslissingsbomen worden in PMML gedefinieerd door middel van een classificatie- of een regressiestructuur. In deze tekst wordt enkel de regressiestructuur behandeld. Elke knoop van een beslissingsboom bevat een logische uitdrukking voor een predikaat of eigenschap die de

regel definieert voor het selecteren van de observaties in de knoop. Een beslissingsboom wordt aangegeven door middel van het element <*TreeModel*> en bezit volgende attributen:

- *modelName*: Identificeert een model door middel van een unieke naam in de context van het PMML document.
- *functionName*: Geeft de functie of het doel weer van het datamining model en is een verplicht attribuut. Mogelijke waarden zijn: associatieregels, reeksen, classificatie, regressie en clustering.
- *algorithmName*: Geeft de naam van het algoritme.
- *splitCharacteristic*: Duidt aan of de beslissingsboom precies twee vertakkingen heeft per knoop (binair) of meerdere vertakkingen.

De voornaamste subelementen van <*TreeModel*> zijn <*MiningSchema*> (zie paragraaf 5.2.3.4) en <*Node*>. Het element *Node* heeft drie kenmerken, namelijk:

- *id*: Dient als een unieke identificator voor iedere knoop van de beslissingsboom.
- *score*: Geeft de voorspelde waarde weer van elk waarneming die wordt gekozen door de knoop (verplicht).
- *recordCount*: Bevat een waarde voor de relatieve grootte van de knoop vergeleken met de ouder knoop.

Voor elke knoop wordt één van de vijf predikatengroepen gedefinieerd:

- <*SimplePredicate*>
  Dit element definieert een regel in de vorm van een eenvoudige waar/onwaar-uitdrukking. Een regel bestaat uit een variabele (uit het mining schema), een binaire vergelijkingsoperator (=, ! =, <, <=, >, >=) en een waarde. De operator kan ook de waarde *isMissing* of *isNotMissing* meekrijgen. In dat geval wordt geen regel gespecificeerd.
- <*CompoundPredicate*>
  Dit element combineert twee of meer elementen die worden gedefinieerd door een predikatengroep. Het attribuut dat wordt geassocieerd met dit element, kan één van de volgende logische operatoren omvatten: *and*, *or*, *xor* of *surrogate*. De operator *and* krijgt de waarde *TRUE* of waar, indien alle predikaten als waar worden geëvalueerd. *Or* heeft de waarde *TRUE*, indien één van de predikaten als waar wordt geëvalueerd. *Xor* wordt enkel toegepast met twee argumenten en krijgt de waarde *TRUE*, indien beide predikaten een verschillende waarde hebben. Tenslotte laat de operator *surrogate* het definiëren van vervangende predikaten toe wanneer een ontbrekende waarde verschijnt in de evaluatie van het ouder predikaat. Hierdoor is een alternatief predikaat beschikbaar.

- *<SimpleSetPredicate>*
  Dit element gaat na of de waarde van een variabele voorkomt als element van een verzameling. Het attribuut dat met dit element wordt geassocieerd, kan één van volgende waarden aannemen: *isIn* of *isNotIn*. De operator *isIn* krijgt de waarde *TRUE* of waar, indien de waarde van de variabele voortkomt in de lijst van waarden. *IsNotIn* krijgt de waarde *TRUE*, indien de waarde van de variabele niet voorkomt in de lijst van waarden.

- *<True>*
  Dit element definieert de constante *TRUE* of waar.

- *<False>*
  Dit element definieert de constante *FALSE* of onwaar.

Indien een waarde ontbreekt in een logische uitdrukking, neemt deze uitdrukking de waarde *UNKNOWN* aan. De manier waarop het *<CompoundPredicate>* element deze waarde evalueert, wordt weergegeven in tabel 5.1.

Tabel 5.1: Evaluatie van onbekende waarden

P	**Q**	**P and Q**	**P or Q**
True	True	True	True
True	False	False	True
True	Unknown	Unknown	True
False	True	False	True
False	False	False	False
False	Unknown	False	Unknown
Unknown	True	Unknown	True
Unknown	False	False	Unknown
Unknown	Unknown	Unknown	Unknown

Deel II

Gevalstudie

Hoofdstuk 6

Analyse van de gegevensstructuur in Lijn BOB

In dit hoofdstuk wordt het ERP-pakket omschreven waarop de gevalstudie is gebaseerd. Achtereenvolgens worden de databasestructuur en de gegevens in de centrale database van het ERP-systeem geanalyseerd. Tenslotte wordt een algemene beoordeling gemaakt over de structuur van de database.

6.1 Lijn BOB

Deze gevalstudie is gebaseerd op een ERP-pakket uit het aanbod van de groep Sage BOB Software. Sage BOB Software is een belangrijke uitgever van boekhoudsoftware, beheerssoftware en CRM-oplossingen voor kleine en middelgrote ondernemingen en fiduciaires op de Belgische en Luxemburgse markt (Sage BOB Software 2005).

Meer concreet hebben we geopteerd voor Lijn BOB (versie 3.0). Dit is een standaard softwarepakket, opgebouwd uit modules, dat zich specifiek richt tot KMO's tot 50 werknemers. De basis van Lijn BOB wordt gevormd door de module Algemene Boekhouding. Aanvullend worden volgende complementaire modules aangeboden:

- Analytische boekhouding: Deze module zorgt voor de opvolging van kostenplaatsen en opbrengsten.
- Commercieel beheer: Deze module behandelt de volledige klanten- en leverancierscyclus van de onderneming; van de bestelling tot de facturatie, van de inventaris tot de levering van leveranciersbestellingen.
- BOB-OLE: Deze module maakt gegevens uit Lijn BOB beschikbaar in Microsoft Excel, Outlook en Word.

- Bankverrichtingen: Deze module biedt onder meer ondersteuning voor internet bankieren.
- Beheer van vaste activa: Deze module behandelt alle verrichtingen die verband houden met de vaste activa.
- Prestatiebeheer: Met behulp van deze module kan men alle dienstverleningen optimaliseren en waarderen (bijvoorbeeld: automatische facturatie van prestaties, beheer van extra kosten, agendabeheer van de medewerkers en de beschikbaarheid van hulpmiddelen).
- E-commerce: Deze module voorziet de mogelijkheid om eenvoudig een website aan te maken en automatisch te beheren, dit alles geïntegreerd in het commercieel beheer.
- BOB-IN: Deze module maakt de overname van gegevens uit andere softwaretoepassingen mogelijk.

6.2 Omschrijving van de informatiebehoefte

De gegevens die worden gebruikt voor het uitwerken van de gevalstudie, zijn gebaseerd op de cijfers van een bestaand bedrijf. Deze werden geregistreerd gedurende een periode van ongeveer vier jaar. Dit maakt dat de gevalstudie een reële bedrijfssituatie weergeeft. Hoewel deze gevalstudie hoofdzakelijk gericht is op het beantwoorden van de centrale onderzoeksvraag en niet rechtstreeks verbonden is aan een onderneming, trachten we alsnog een realistische situatie na te bootsen. We veronderstellen daarom dat we meer informatie wensen te bekomen over de winstgevendheid van klanten op basis van de klantaankopen die werden geregistreerd met behulp van de module Commercieel Beheer in Lijn BOB.

Door het uitvoeren van een analyse van de gegevens en de databasestructuur van Lijn BOB, trachten we enerzijds te bepalen welke gegevens nodig zullen zijn om de gewenste informatie te bekomen. Anderzijds gaan we na of de databasestructuur van Lijn BOB geschikt is voor toepassing van een Business Intelligence applicatie, namelijk datamining.

6.3 Analyse van de databasestructuur

De verschillende modules in Lijn BOB zijn opgebouwd rond een gemeenschappelijke database. Dit is tevens één van de voornaamste eigenschappen van een ERP-systeem. Lijn BOB maakt gebruik van een relationele database, namelijk Paradox van Borland Software Corporation. Indien men alle modules van Lijn BOB installeert, bevat deze database meer dan 180 verschillende tabellen. Om in deze complexe structuur meer overzicht te krijgen, wordt aan elke tabel een type toegekend. De voornaamste types zijn:

- Company (S): Tabellen die betrekking hebben op de algemene boekhouding
- General (G): Tabellen die betrekking hebben op het prestatiebeheer
- Fiscal (F): Tabellen die betrekking hebben op de BTW-aangifte
- Invoice (I): Tabellen die betrekking hebben op het commercieel beheer
- Fixed Assets (FA): Tabellen die betrekking hebben op de vaste activa

Aangezien het onmogelijk is om alle tabellen gedetailleerd te analyseren, beperken we ons tot één specifieke module die ook in de verdere uitwerking van de gevalstudie centraal zal staan: het Commercieel Beheer. Alle gegevens die met behulp van deze module worden verzameld, worden opgeslagen in tabellen van het type Invoice (I). Tabel 6.1 geeft een overzicht van alle tabellen van dit type. Een belangrijke gegevenstabel die alsnog ontbreekt, is de tabel met klantgegevens. Deze vinden we terug onder het type Company (S), namelijk de tabel COMPAN.

Tabel 6.1: Tabellen van het type Invoice

Tabel	**Gegevensinhoud**	**Kolommen**
CUCCTC	bevat informatie met betrekking tot contracten	25
EDTCFG	bevat configuratiegegevens	16
IART	bevat algemene gegevens van ieder artikel	128
IARTSP	bevat gegevens over de leveranciers en aankoopprijzen van een artikel	13
IARTT	bevat voorraadgegevens van ieder artikel	15
IASART	bevat informatie over de componenten waaruit een artikel is samengesteld	8
ICTC	bevat extra informatie met betrekking tot contracten	26
ICTLG	bevat informatie over de artikelcatalogus	6
IDISCT	bevat informatie over de kortinggroepen waartoe artikels behoren	8
IDOCT	bevat informatie over de factuurtotalen	25
IDORD	bevat gegevens voor het linken van het klantorder en het leveranciersorder	22
IHDDOC	bevat informatie over de factuurhoofdingen	107
IHIDN	bevat informatie met betrekking tot de levering van goederen	77
IHISTO	bevat een historiek van artikelbewegingen	77
ILDEF	bevat instellingen voor het afdrukken van documenten	21
IMSG	bevat toegevoegde boodschappen	14
INVCFG	bevat de parameters van de module Commercieel Beheer	185
IPRLIS	bevat gegevens over de prijslijsten	8
IPRVER	bevat gegevens over de geldige periodes van de prijslijsten	12
ISTATI	bevat statistieken met betrekking tot artikels	12
ITMPLT	afgeleide van de tabel IHISTO	63
IWAREH	bevat gegevens over de voorraden en de magazijnen	12

6.4 Analyse van de gegevens

Bij de analyse van de tabelstructuur beperken we ons tot die tabellen die nuttige gegevens bevatten met betrekking tot de vooropgestelde informatiebehoefte. Tabellen die zijn gerelateerd aan voorraden, prijslijsten en catalogi worden daarom niet beschouwd. Daarnaast maken we volgende vereenvoudigende assumpties:

1. Elk artikel wordt aangekocht bij één leverancier en heeft bijgevolg slechts één aankoopprijs.
2. Er worden geen samengestelde artikels aangeboden.
3. Er worden geen kortingen toegekend.
4. BTW wordt niet in rekening gebracht.
5. Het klantenbestand bevat enkel en alleen Belgische klanten.

Op die manier reduceren we het aantal tabellen tot IART, IHISTO en COMPAN. Deze tabellen worden vervolgens individueel behandeld.

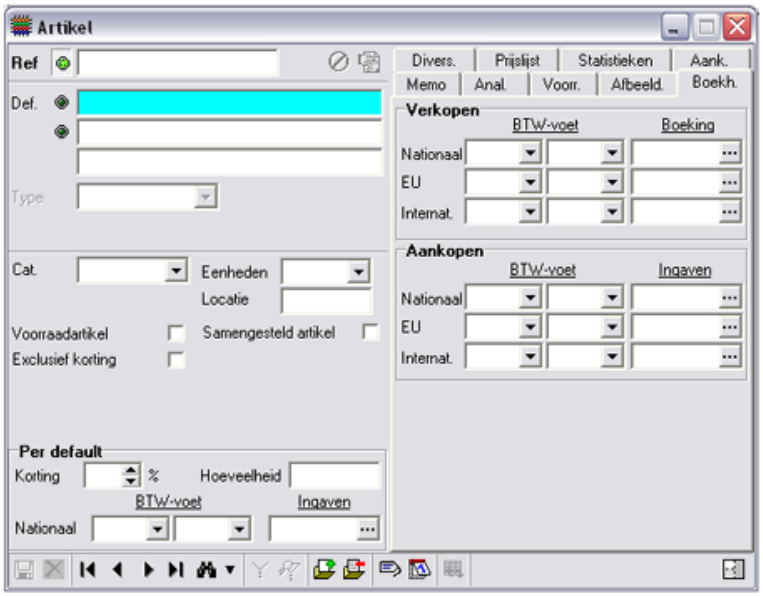
6.4.1 De tabel IART

De tabel IART omvat een gedetailleerde omschrijving van elk artikel dat door de onderneming wordt aangeboden. Van elk artikel worden maar liefst 128 kenmerken bijgehouden. De meeste van deze attributen worden door de gebruiker ingevuld op de artikelfiche in Lijn BOB (figuur 6.1). Hierna volgt een opsomming van de voornaamste velden in de tabel IART. Een volledig overzicht is opgenomen in bijlage A.

- NUM: Een uniek intern nummer dat automatisch wordt toegekend aan elk artikel.
- REF: Een unieke referentie die door de gebruiker wordt opgegeven voor een artikel.
- HEADING1: Een algemene omschrijving van het artikel.
- CAT: De categorie waartoe het artikel behoort.
- PURPRICE: De aankoopprijs van het artikel.

6.4.2 De tabel COMPAN

De tabel COMPAN bevat alle gegevens die betrekking hebben op klanten, leveranciers en prospecten. Elke klant/leverancier/prospect heeft 83 eigenschappen. Ook deze attributen worden door de gebruiker ingevuld op een fiche in Lijn BOB. De klantenfiche wordt weergegeven in



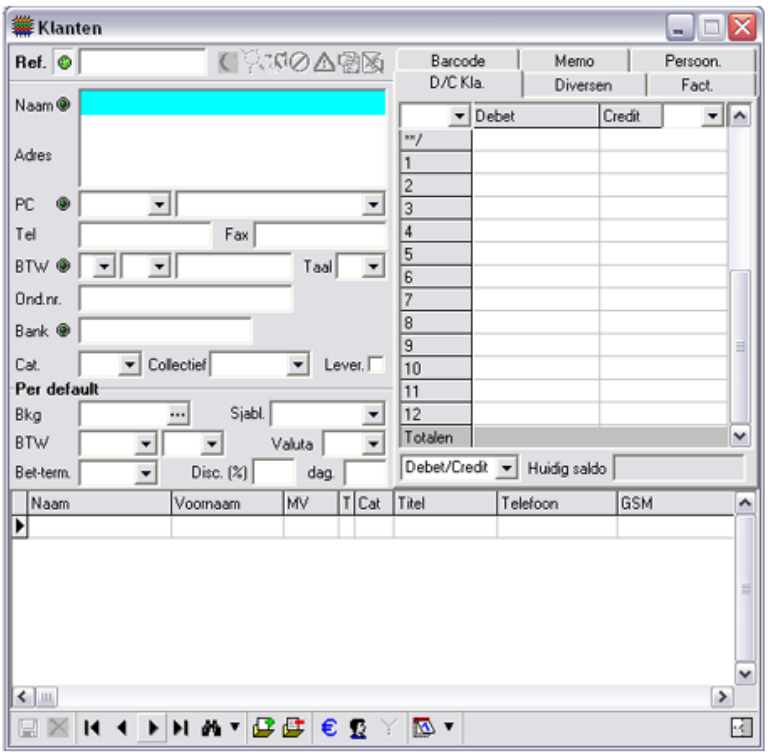
Figuur 6.1: Artikelfiche in Lijn BOB

figuur 6.2. Een volledig overzicht van de kolommen van de tabel COMPAN is terug te vinden in bijlage B. Deze tekst beperkt zich tot het opsommen van de belangrijkste attributen.

- CID: Een unieke referentie die door de gebruiker wordt opgegeven voor een klant.
- CCUSTYPE: Geeft aan of de derde al dan niet een klant is.
- CNAME1: De naam van de klant.
- CADDRESS1: Het adres van de klant.
- CZIPCODE: De postcode van de klant.
- CLOCALITY: De plaats waar de klant gevestigd is.
- CVATNO: Het BTW-nummer van de klant.
- CTELNO: Het telefoonnummer van de klant.
- CFAXNO: Het faxnummer van de klant.

6.4.3 De tabel IHISTO

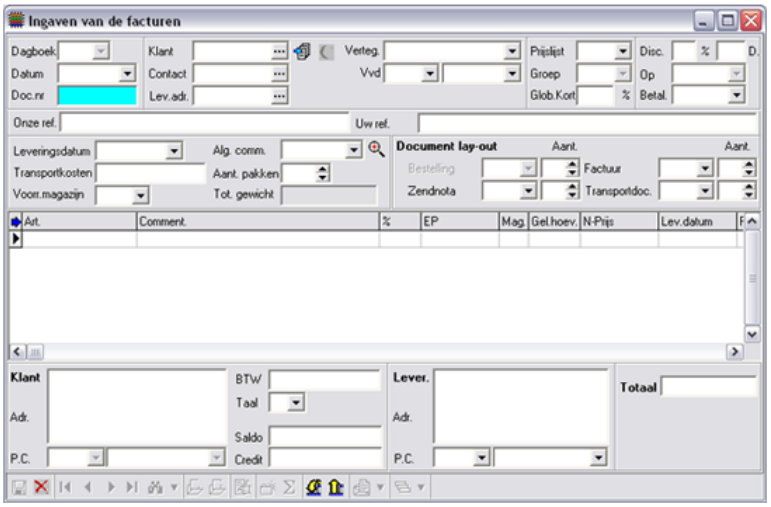
De tabel IHISTO tenslotte, legt het verband tussen artikels en klanten door het registreren van de artikelbewegingen. Deze tabel bevat de detaillijnen van alle ingegeven documenten, inclusief de verkoopfacturen. Elke lijn van een document wordt opgeslagen door middel van



Figuur 6.2: Klantenfiche in Lijn BOB

77 attributen. Het ingeven van verkoopfacturen gebeurt op basis van het scherm dat getoond wordt in figuur 6.3. Hierna volgt een overzicht van de voornaamste velden uit de tabel IHISTO. Voor een overzicht van de volledige tabel, verwijzen we naar bijlage C.

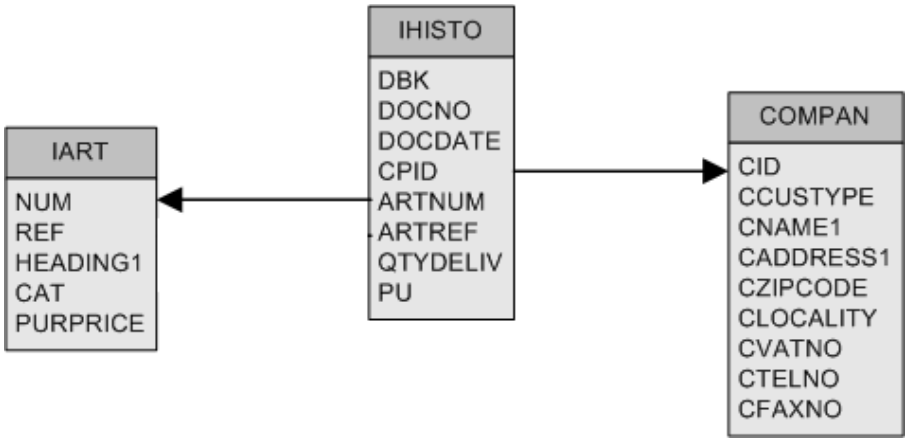
- DBK: De code van het dagboek.
- DOCNO: Het unieke nummer van de factuur dat wordt toegekend door de gebruiker.
- DOCDATE: De factuurdatum.
- CPID: De unieke referentie van de klant.
- ARTNUM: Het uniek nummer van het artikel.
- ARTREF: De unieke referentie van het artikel.
- QTYDELIV: Het aantal verkochte eenheden van een artikel.
- PU: De verkoopprijs van het artikel.



Figuur 6.3: Ingaven van de facturen in Lijn BOB

6.4.4 De relaties tussen IART, COMPAN en IHISTO

De relaties tussen de tabellen IART, COMPAN en IHISTO worden weergegeven in figuur 6.4. De tabellen IART en IHISTO zijn verbonden door middel van een één-op-veel relatie, meer bepaald tussen de velden NUM (IART) en ARTNUM (IHISTO). Elke factuurlijn refereert immers naar precies één artikel. Een artikel daarentegen kan in meerdere factuurlijnen voorkomen. Een analoge relatie bestaat tussen de tabel COMPAN en IHISTO. Elk verkocht artikel behoort tot precies één klant, terwijl een klant meerdere artikels kan kopen. Het veld CPID uit de tabel IHISTO komt dus overeen met het veld CID uit de tabel COMPAN.



Figuur 6.4: Onderlinge relaties tussen de tabellen IART, COMPAN en IHISTO

6.5 Beoordeling van de databasestructuur

In de theoretische studie werd meermaals de aandacht gevestigd op de beperkingen van operationele systemen ten aanzien van de ondersteuning van analyses. Na het bestuderen van de Paradoxdatabase van Lijn BOB, kunnen we vaststellen of deze beperkingen van toepassing zijn op dit ERP-systeem.

Een eerste beperking van ERP-databases is het gebrek aan historische gegevens die meestal noodzakelijk zijn voor het uitvoeren van analyses. Dit is echter niet geheel van toepassing op Lijn BOB, aangezien een onbeperkt aantal boekjaren kan worden bijgehouden. Toch moet hierbij een kanttekening worden gemaakt. Het bijhouden van gegevens over een groot aantal periodes wordt niet aangeraden omdat dit de prestaties van het systeem negatief zou beïnvloeden. Lijn BOB beschikt daarom over een speciale archiveringsfunctie, waardoor een deel van de gegevens worden overgebracht naar een andere map of een verplaatsbare eenheid. Dit laat de mogelijkheid open voor de gebruiker om de gearchiveerde historieken te raadplegen. In de huidige versie van Lijn BOB is deze functie enkel beschikbaar voor de algemene boekhouding. Een beperking die echter wel van toepassing is op de database van Lijn BOB is dat enkel interne gegevens worden verzameld en geregistreerd.

Een tweede belangrijk nadeel van operationele systemen is de onstabiele omgeving die wordt gecreëerd door het continu aanbrengen van wijzigingen. Dit bemoeilijkt het uitvoeren van betrouwbare analyses. Ook Lijn BOB is een transactiesysteem dat onderhevig is aan voortdurende veranderingen.

Een derde beperking van de databasestructuur, is de manier waarop de gegevens zijn ingedeeld. Voor het uitvoeren van analyses, is een indeling op basis van bedrijfsonderwerpen (klant, product, verkoop, tijd, ...) gewenst. De tabelstructuur van Lijn BOB stemt niet overeen met deze indeling. De gegevensinhoud van de tabel IHISTO geeft bijvoorbeeld aan dat geen duidelijk onderscheid wordt gemaakt tussen het bedrijfsonderwerp verkoop en de overige artikelbewegingen (offertes, bestellingen, zendnota's, ...).

Tenslotte brengt ook de complexiteit van de databasestructuur enkele nadelen met zich mee. De complexiteit wordt voornamelijk veroorzaakt door de grote hoeveelheid tabellen. Dit is het resultaat van het normaliseren van tabellen met als doel het vermijden van redundante gegevens. De complexe databasestructuur heeft als grootste nadeel dat men zeer ingewikkelde query's moet construeren om gegevens uit de database op te vragen en dat bijgevolg de responstijd van deze query's aanzienlijk toeneemt. In Lijn BOB vinden we een zeer groot aantal tabellen terug als gevolg van normalisatie. Toch zijn er enkele kleine uitzonderingen. Dit wordt

geïllustreerd aan de hand van een voorbeeld: De derdenreferentie (CPID) wordt zowel op het niveau van de detaillijnen in de tabel IHISTO als op het niveau van de factuurhoofdingen in de tabel IHDDOC opgeslagen. Hierdoor wordt eenzelfde gegeven op meerdere plaatsen opgeslagen en ontstaat redundantie. Een mogelijke verklaring is de vereenvoudiging die wordt bekomen in de query's voor het opvragen van gegevens uit de tabel IHISTO. In plaats van drie tabellen (IHISTO, IHDDOC, COMPAN) aan te spreken voor het opvragen van de verkopen van een specifieke klant, hoeft men nu slechts twee tabellen op te vragen (IHISTO, COMPAN). Dit toont aan dat de complexiteit van de query's als gevolg van de vele tabellen daadwerkelijk een probleem kan vormen bij ERP-systemen.

Andere vaak voorkomende nadelen zoals vervuiling en vergrendeling werden niet vastgesteld in de Paradoxdatabase. Doordat we in deze gevalstudie slechts gebruik maken van één gegevensbron, was het onmogelijk om beperkingen te onderzoeken met betrekking tot verschillende technische infrastructuren, ongelijksoortige databasestructuren en gefragmenteerde gegevens.

Na het analyseren van de beperkingen, stellen we vast dat de databasestructuur van Lijn BOB niet optimaal is voor het ondersteunen van een Business Intelligence applicatie. Om een beter afgestemde databasestructuur te bekomen, zullen we een afzonderlijke database construeren die een betrouwbare basis zal vormen voor toepassing van datamining.

Hoofdstuk 7

Constructie van een datamart met behulp van ETL4ALL

In hoofdstuk zes werd aangetoond dat de gegevensstructuur van Lijn BOB geen voldoende basis biedt voor het uitvoeren van analyses. We trachten daarom in dit hoofdstuk, met behulp van een ETL-toepassing, de gegevens in Lijn BOB te extraheren, transformeren en nadien in een meer geschikte structuur om te zetten. In het eerste deel ontwikkelen we een dimensionaal gegevensmodel dat de basis zal vormen voor een datamart. In de volgende secties construeren we de datamart met behulp van de softwaretoepassing ETL4ALL. Tenslotte leiden we een aantal variabelen af die de input zullen vormen van een datamining model.

7.1 Ontwerp van het dimensionaal gegevensmodel

De ontwikkeling van een datamart wordt gebaseerd op een model. Een goede methode voor het ontwerpen van dit model is dimensionaal modelleren. In deze gevalstudie passen we de vier-stappen-methodologie toe van de grondlegger van het dimensionaal modelleren, namelijk Ralph Kimball, om een geschikt gegevensmodel uit te werken.

7.1.1 Stap 1: Het onderwerp selecteren van de datamart

De eerste stap in het ontwerp bestaat uit het selecteren van het bedrijfsproces dat zal worden gemodelleerd. Deze beslissing kan enkel worden genomen na een grondige analyse van de beschikbare gegevens en een algemeen inzicht in de informatiebehoeften van de onderneming. In hoofdstuk zes werd reeds de nodige aandacht besteed aan de structuur en de inhoud van de Paradoxdatabase van Lijn BOB. Ook werd bepaald dat we meer informatie wensen te bekomen over de klantaankopen die worden geregistreerd met behulp van de module Commercieel Beheer in Lijn BOB. Het bedrijfsproces dat we wensen te modelleren is bijgevolg het verkoopsproces.

Met behulp van de verkoopsgegevens trachten we de winstgevendheid van klanten in kaart te brengen.

7.1.2 Stap 2: Het aangeven van de betekenis van de feitentabel

Nadat het bedrijfsproces werd geïdentificeerd, moet een belangrijke beslissing worden genomen over de betekenis van elke rij in de feitentabel. In Lijn BOB vinden we de meest atomaire gegevens in het verkoopsproces terug als een individuele lijn op een verkoopfactuur. Om een maximale dimensionaliteit en flexibiliteit te garanderen, baseren we de feitentabel op dit gegeven.

7.1.3 Stap 3: De dimensies identificeren en conformeren

Na het vastleggen van de betekenis van de feitentabel, ligt de keuze van de dimensies voor de hand. Een detaillijn van een verkoopfactuur bevat een verwijzing naar een bepaald *product*. Dit is de eerste dimensie. De voornaamste attributen die een product beschrijven zijn: de referentie, de omschrijving en de productcategorie. Als tweede dimensie kiezen we voor *klant* aangezien deze de aankoop van een product tot stand brengt. Een klant wordt omschreven op basis van zijn naam, adres, postcode en gemeente. Verder behoort een detaillijn tot een bepaalde *factuur* met een uniek factuurnummer. Dit is de derde dimensie. Tenslotte voegen we een *tijdsdimensie* toe, die ons toelaat om aankopen te relateren aan een bepaald tijdstip. Als attributen kiezen we voor: dag van de week, dag van het jaar, week van het jaar, maand, kwartaal en jaar. We komen zo tot een totaal van vier dimensies. Deze hebben elk een unieke sleutel die deel uitmaakt van de samengestelde sleutel van de feitentabel. Aangezien we voorlopig slechts één datamart construeren, is het conformeren van de dimensies niet nodig.

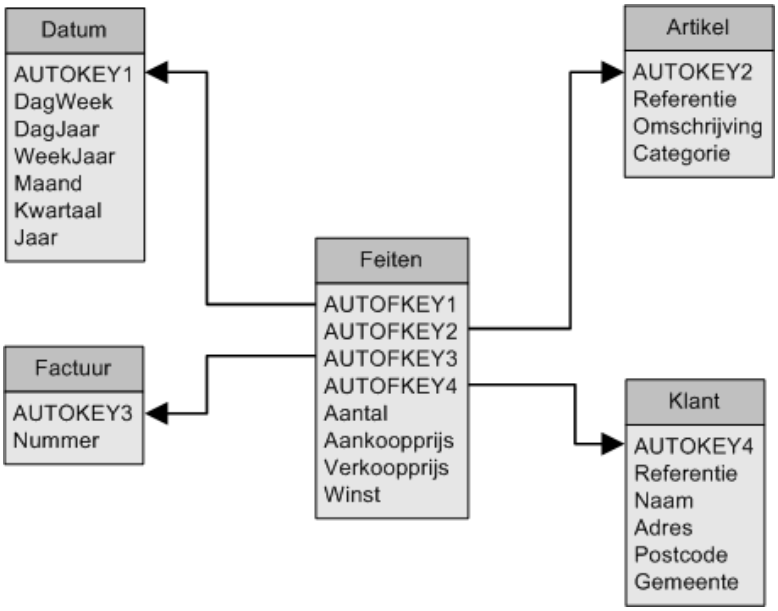
7.1.4 Stap 4: De feiten kiezen

In de vierde en laatste fase van het ontwerp, moet een voorzichtige bepaling worden gemaakt van de feiten die in de feitentabel zullen verschijnen. Deze feiten moeten overeenstemmen met de betekenis die aan de feitentabel werd toegekend in stap twee, met name de detaillijnen van een factuur. De feiten die door middel van de ingegeven verkoopfacturen worden verzameld in Lijn BOB zijn: het aantal verkochte producten, de aankoopprijs, de verkoopprijs en de winst. De aankoopprijs wordt bepaald door het vermenigvuldigen van het aantal verkochte eenheden met de aankoopprijs van een individueel product. Een analoge berekening geldt voor de verkoopprijs. De winst wordt berekend door het verminderen van de verkoopprijs met de aankoopprijs. Hoewel de winstberekening erg eenvoudig is, is het toch zinvol om deze fysiek op te slaan in de feitentabel. Aangezien de winst op diverse manieren kan worden berekend, voorkomen we zo dat eindgebruikers zich op verschillende winstcijfers zouden baseren.

Daarnaast is het winstcijfer een erg belangrijk gegeven dat waarschijnlijk vaak in de analyse zal terugkeren. We willen dan ook vermijden dat het meermaals herberekenen van de winst een ongunstig effect zou hebben op de responstijd van zoekopdrachten.

7.1.5 De ster

Het sterschema dat wordt bekomen met behulp van Kimball's methode, levert volgende visuele voorstelling op:



Figuur 7.1: Het sterschema van het verkoopsproces

7.2 ETL4ALL

Voor de constructie van een datamart, stelde de onderneming IKAN Software één van haar producten ter beschikking, namelijk de toepassing ETL4ALL. IKAN Software is een in België gevestigde internationale onderneming, die zich hoofdzakelijk richt op het creëren van toepassingen voor softwareontwikkeling.

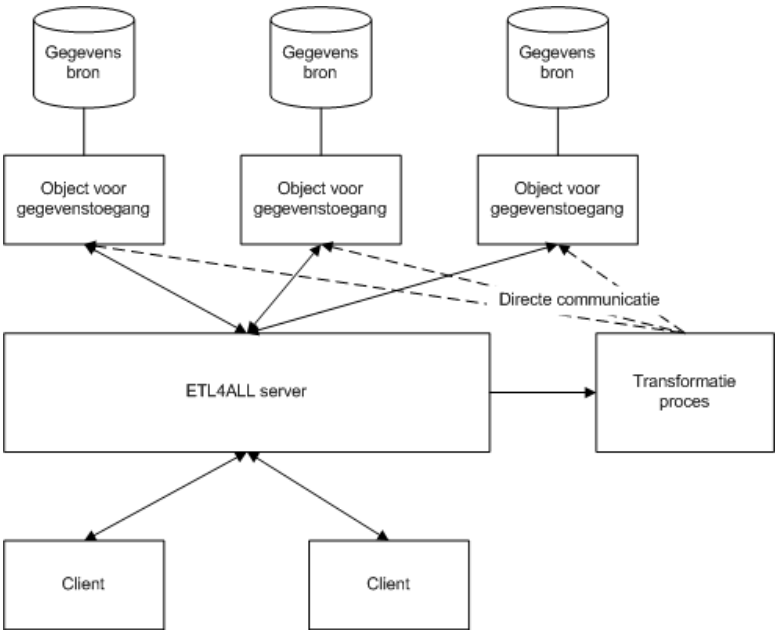
7.2.1 Het doel van ETL4ALL

Business Intelligence (BI) toepassingen spelen steeds vaker een centrale rol in het strategische en tactische management van een organisatie. Ze kunnen verschillende vormen aannemen, maar zijn steeds afhankelijk van de onderliggende verzameling van gegevens. Deze basis wordt gevormd door het extraheren, transformeren en laden van gegevens uit diverse bronnen in de

BI-applicatie. Dit proces is echter tijdrovend en complex en telt mee voor meer dan 60 procent van de kosten. Om dit proces op een efficiënte manier te doorlopen, kan men gebruik maken van ETL4ALL. ETL4ALL creëert een solide gegevensbasis die essentieel is voor elke BI-toepassing. Het biedt de mogelijkheid aan eindgebruikers om gegevens uit een willekeurige bronomgeving te benaderen, te transformeren en vervolgens in een doelomgeving te laden (ETL4ALL 2005).

7.2.2 De architectur van ETL4ALL

Figuur 7.2 toont de verschillende componenten van de ETL4ALL architectuur (IKAN-Software 2004).



Figuur 7.2: ETL4ALL architectuur

- ***Gegevensbron:*** ETL4ALL ondersteunt de meest voorkomende gegevensbronnen zoals relationele databases, XML databases, PDF, SAP, ...
- ***Object voor gegevenstoegang:*** Een specifiek object zorgt voor de communicatie tussen een gegevensbron en de centrale ETL4ALL server. Hierbij wordt een standaard interface geïmplementeerd die onafhankelijk is van het type gegevensbron dat wordt ondersteund.
- ***ETL4ALL server:*** De ETL4ALL server vormt de kern van het ETL4ALL proces:
  - De server verzamelt metadata van de beschikbare gegevensbronnen.
  - De server ontvangt metagegevens van de ETL4ALL clients en slaat deze op in de centrale bewaarplaats.

  - De server ontvangt transformatieopdrachten van ETL4ALL clients.
  - De server interpreteert deze transformatieopdrachten en zendt ze naar het transformatieproces.
  - De server zorgt ervoor dat de overeenstemmende gegevensresultaten ontvangen worden door de ETL4ALL client.

- ***Transformatieproces:*** Het transformatieproces is een afzonderlijk proces op de ETL4AL server dat de gegevenstransformaties uitvoert:
  - Het proces ontvangt opdrachten voor gegevenstransformaties van de ETL4ALL server.
  - Het proces verzamelt de gegevens die nodig zijn voor de transformaties rechtstreeks van de gegevensbronnen of van een ander transformatieproces (indien de gegevens resulteren uit een combinatie van transformaties).
  - Het proces voert de gewenste gegevenstransformaties uit.
  - Het proces stuurt de resultaten naar de ETL4ALL server die ze bezorgt aan de client. OF
  - Het proces stuurt de resultaten naar een ander transformatieproces in geval van gecombineerde transformaties.
  - Het proces stuurt de resultaten naar een door de gebruiker gedefinieerde gegevensbron in een standaard XML-formaat.
  - De resultaten worden tenslotte in de aangewezen gegevensbron geladen.

- ***Client:*** ETL4ALL beschikt over een standaard client-toepassing.

7.3 Opvulling van de datamart met behulp van ETL4ALL

In het eerste deel van dit hoofdstuk werd een model ontworpen voor een datamart die we wensen op te vullen met de gegevens uit de database van Lijn BOB. In dit deel van het hoofdstuk worden de gegevens uit Lijn BOB effectief getransformeerd en in de datamart geladen met behulp van ETL4ALL.

7.3.1 Inleiding

In deze algemene inleiding worden enkele begrippen en technieken besproken die bepalend zijn voor de werkwijze in ETL4ALL. Deze inleiding werd samengesteld aan de hand van de gebruikershandleiding van ETL4ALL (IKAN-Software 2004).

Een gegevenstransformatie wordt uitgevoerd in drie fasen:

1. Het analyseren van de behoeften.
2. Het definiëren van een transformatieprogramma.
3. Het uitvoeren van een transformatieprogramma.

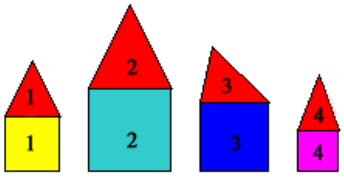
Deze fasen worden hierna meer gedetailleerd behandeld.

7.3.1.1 Het analyseren van de behoeften

De analysefase bestaat uit:

- Het definiëren van de vereiste doelgegevens.
- Het traceren van de benodigde brongegevens om deze doelgegevens te bekomen.
- Het bepalen van de transformatie of de opeenvolging van transformaties die moeten worden uitgevoerd om de vereiste doelgegevens te bekomen uit de beschikbare brongegevens.

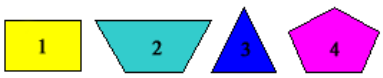
Deze principes worden toegepast aan de hand van een eenvoudig voorbeeld. We veronderstellen als doelgegevens een verzameling huizen waarbij elk huis is samengesteld uit een vierkant (muren) en een rode driehoek (dak) (figuur 7.3).



Figuur 7.3: Doelgegevens: huizen

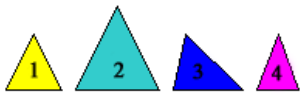
Verder beschikken we over twee gegevensbronnen om deze huizen te bouwen namelijk:

- Een eerste gegevensbron die veelhoeken bevat in alle kleuren (figuur 7.4).



Figuur 7.4: Gegevensbron: veelhoeken

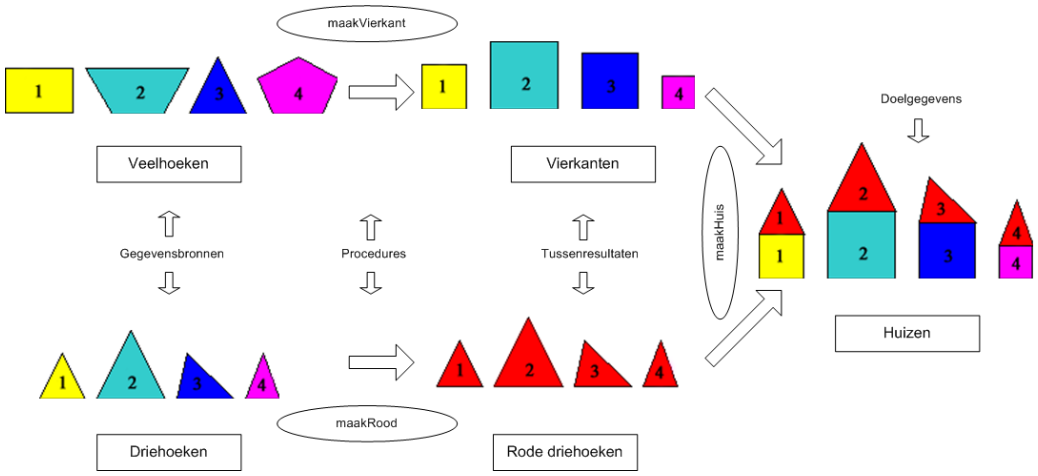
- Een tweede gegevensbron die driehoeken bevat in alle kleuren (figuur 7.5).



Figuur 7.5: Gegevensbron: driehoeken

We moeten vervolgens bepalen hoe we deze brongegevens zullen transformeren naar de doelgegevens. Deze transformatie kan niet rechtstreeks worden uitgevoerd, maar zal bestaan uit drie stappen (figuur 7.6):

- De driehoeken moeten een rode kleur krijgen.
- De veelhoeken moeten worden omgevormd tot vierkanten.
- De resulterende driehoeken en vierkanten moeten worden gecombineerd om een huis te bouwen.



Figuur 7.6: Transformaties voor het voorbeeld met huizen

7.3.1.2 Het definiëren van een transformatieprogramma

Metagegevens

Een belangrijke eigenschap van ETL4ALL is dat het zo lang mogelijk streeft naar het werken met metagegevens in plaats van actuele data. Een metadata definitie is vergelijkbaar met de definitie van een databasetabel. Daar waar de definitie van een databasetabel een opsomming

geeft van een aantal kolommen die de kenmerken beschrijven van elke rij, geeft een metadata definitie een overzicht van een aantal velden met hetzelfde doel. Pas wanneer een transformatie-programma effectief wordt uitgevoerd, zullen de metadata definities worden geassocieerd met de gegevensbronnen. Metagegevens moeten daarom worden gedefinieerd voor elke gegevensbron, voor elk tussenresultaat en voor de eindbestemming van de gegevens. In het voorbeeld geeft dit:

- Driehoeken (ID, grootte, kleur)
- Rode driehoeken (ID, grootte, kleur)
- Veelhoeken (ID, grootte, aantal hoeken)
- Vierkanten (ID, grootte, aantal hoeken)
- Huizen (ID, grootte)

Elke gegevensbron die wordt geselecteerd voor het uitvoeren van de gegevenstransformatie, moet een structuur hebben die overeenkomt met de metadata definitie.

Procedures

Procedures definiëren de bewerkingen die nodig zijn om de brongegevens te transformeren naar de tussenresultaten of naar de doelgegevens. In het voorbeeld hebben we drie procedures nodig:

- MaakRood: Deze procedure verandert de kleur van elke driehoek in rood. Het resultaat van deze procedure is de metadata definitie Rode driehoeken.
- MaakVierkant: Deze procedure vormt elke veelhoek om naar een vierkant door het aantal hoeken te veranderen in vier. Deze procedure resulteert in de metadata definitie Vierkanten.
- MaakHuis: Deze procedure combineert het vierkant en de driehoek met hetzelfde ID en vormt zo een huis met dit ID waarbij de respectievelijke groottes worden opgeteld. Deze procedure levert de metadata definitie Huizen op.

Programma's

Een programma bestaat uit een opeenvolging van procedures die nodig zijn voor het uitvoeren van de gegevenstransformatie. Programma's worden geassocieerd met de uiteindelijke doelgegevens. Het programma dat behoort bij het voorbeeld van de huizen, wordt getoond in figuur 7.6.

Gegevensbronnen

Na het ontwikkelen van een programma, moet men twee gegevensbronnen definiëren of selecteren:

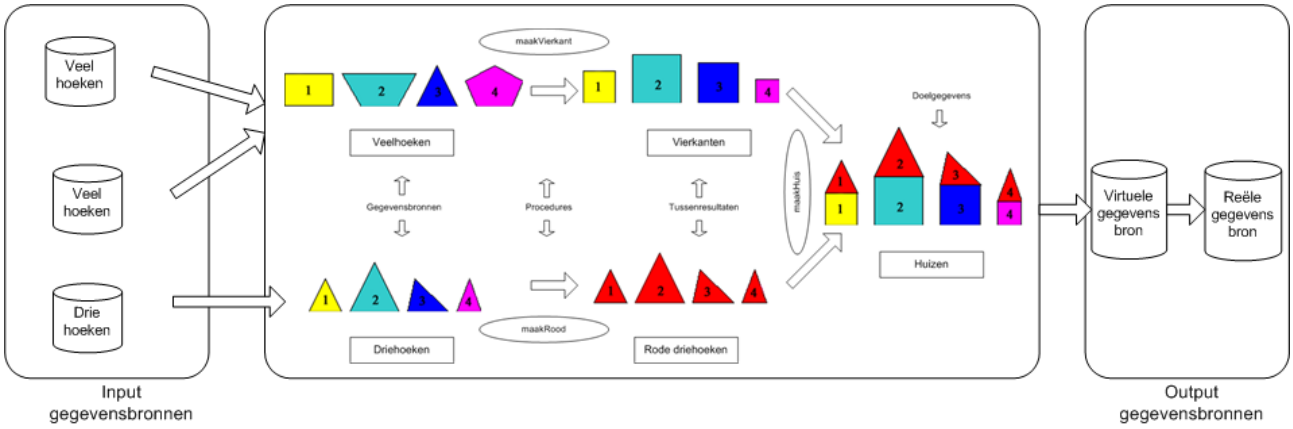
- *Input gegevensbronnen*
  Dit zijn gegevensbronnen die de actuele brongegevens van het programma bevatten. Input gegevensbronnen verwijzen altijd naar bestaande, fysiek aanwezige bestanden of databasetabellen. Dit noemt met ook wel reële gegevensbronnen.

- *Output gegevensbronnen*
  Dit zijn gegevensbronnen die de output of het resultaat van het programma zullen bevatten. Er bestaan twee typen output gegevensbronnen namelijk:

  – Virtuele gegevensbronnen: De resultaten van het programma worden niet opgeslagen op een fysieke locatie. De resultaten worden ad hoc berekend en op het scherm weergegeven, maar ze gaan verloren van zodra men het venster afsluit. Het is wel mogelijk deze gegevens als input te gebruiken voor een ander programma of om te zetten naar een reële gegevensbron.

  – Reële gegevensbronnen: De resultaten van het programma worden fysiek opgeslagen. Nadien kan men ze steeds raadplegen.

Figuur 7.7 duidt de verschillende gegevensbronnen aan in het voorbeeld van de huizen.



Figuur 7.7: Gegevensbronnen voor het voorbeeld met huizen

7.3.1.3 Het uitvoeren van een transformatieprogramma

De uitvoering van een transformatieprogramma wordt manueel gestart door de gebruiker.

7.3.2 Stap 1: het importeren van de gegevens in ETL4ALL

In de eerste stap van het transformatieproces, importeren we zowel de gewenste doelgegevens als de brongegevens die nodig zijn om deze doelgegevens te bekomen. Daarnaast bepalen we de transformaties om de brongegevens om te vormen naar de gewenste doelgegevens.

Het doel van het transformeren van de gegevens in Lijn BOB is het opvullen van een datamart. Deze datamart bepaalt dus de doelgegevens van het transformatieproces. Het dimensionaal gegevensmodel dat beschreven is aan het begin van dit hoofdstuk, moet worden omgezet naar een fysieke databasestructuur (bijlage D bevat de SQL-code waarmee de database werd gecreëerd). Deze database bestaat uit volgende tabellen (de kolommen worden tussen haakjes vermeld):

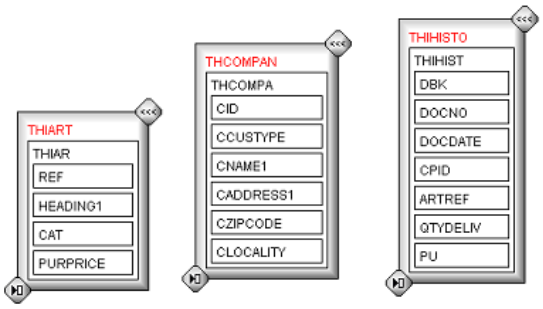
- DATUM(AUTOKEY1, DagWeek, DagJaar, WeekJaar, Maand, Kwartaal, Jaar)
- ARTIKEL(AUTOKEY2, Referentie, Omschrijving, Categorie)
- FACTUUR (AUTOKEY3, Nummer)
- KLANT (AUTOKEY4, Referentie, Naam, Adres, Postcode, Gemeente)
- FEITEN (AUTOFKEY1, AUTOFKEY2, AUTOFKEY3, AUTOFKEY4, Aantal, Aankoopprijs, Verkoopprijs, Winst)

We importeren de tabellen Datum, Artikel, Factuur, Klant en Feiten in ETL4ALL. Om deze datamart op te vullen, moeten we beschikken over de gegevens uit de database van Lijn BOB. We importeren daarom de drie voornaamste tabellen: IART, COMPAN en IHISTO. Na het importeren van de benodigde tabellen uit de Paradoxdatabase, openen we de grafische voorstellingen van de metadata definities van deze tabellen. We stellen daarbij vast dat de Paradoxtabellen heel wat kolommen bevatten die niet nodig zijn voor het opvullen van de datamart. Een eerste taak bestaat er dan ook in om de overbodige kolommen te schrappen. Dit doen we eenvoudigweg door het wissen van de overeenkomstige kolomnamen uit de metadata definities. Het resultaat wordt weergegeven in figuur 7.8.

Na het bestuderen van de bron- en doelgegevens stellen we vast dat volgende transformaties nodig zijn om de brongegevens om te vormen naar de doelgegevens:

1. De gegevens in de tabellen IART, COMPAN en IHISTO moeten worden gecombineerd, zodat enkel de gegevens die gerelateerd zijn aan een verkoop, worden opgenomen.
2. Enkel detaillijnen van de verkoopfacturen mogen worden geëxtraheerd uit de tabel IHISTO.
3. Alleen klanten mogen worden overgebracht uit de tabel COMPAN.

4. Detaillijnen die geen betrekking hebben op een verkoop moeten worden verwijderd.
5. De factuurdatum moet worden vertaald in dag van de week, dag van het jaar, week van het jaar, maand, kwartaal en jaar.
6. De aankoopprijs en de verkoopprijs moeten worden berekend.
7. De winst moet worden berekend aan de hand van de aankoop- en verkoopprijs.



Figuur 7.8: Grafische voorstelling van de metadata definities van IART, COMPAN en IHISTO in ETL4ALL

Aan het einde van de eerste fase, beschikken we over de nodige brongegevens en een lege datamart. In de volgende stap worden de transformaties omgezet naar enkele concrete procedures.

7.3.3 Stap 2: Het definiëren van procedures

In stap twee vertalen we de transformaties die nodig zijn om de brongegegevens om te vormen naar de doelgegevens, in enkele concrete procedures. Eerst voegen we manueel een nieuwe metadata definitie toe die we *ster* noemen. Deze definitie is gebaseerd op het sterschema dat aan de basis ligt van de datamart en bevat een kopie van de metadata definities van de feitentabel en de dimensies. Deze ster zal bij het definiëren van de procedures zeer handig zijn.

ETL4ALL vereenvoudigt het maken van procedures doordat het een grote hoeveelheid voorgedefinieerde transformaties bevat. Met behulp van deze transformaties maken we twee procedures aan:

- ***vulDatamart*** (figuur 7.9)
  In deze procedure worden de meeste transformaties uitgevoerd. We vertrekken van de metadata definities van de tabellen IART, IHISTO en COMPAN (bronnen) en zullen door middel van een aantal functies de metadata definitie van de *ster* (doel) trachten op te vullen. Volgende bewerkingen worden daarvoor uitgevoerd:

1. *Het samenvoegen van de tabellen IART, IHISTO en COMPAN*
   Indien een procedure verwijst naar meer dan één bron metadata definitie, dan moet worden gedefinieerd hoe de rijen van deze bronnen aan elkaar worden gekoppeld. Hiervoor voorziet ETL4ALL de relationele functie *Join*. De matching gebeurt op basis van een *Join Field* voor elke bron. Indien de rijen die behoren tot verschillende bronnen een identieke waarde hebben voor dit *Join Field*, dan worden ze samen beschouwd bij het produceren van een doelrij. Aangezien we voor deze procedure over drie metadata definities beschikken als bron, maken we gebruik van twee *Join*-functies. De eerste *Join* voegt de tabel IART samen met de tabel IHISTO op basis van de velden *REF* (IART) en *ARTREF* (IHISTO). De tweede functie voegt de tabel IHISTO samen met de tabel COMPAN op basis van de velden *CPID* (IHISTO) en *CID* (COMPAN).
2. *Het elimineren van onbruikbare gegevens*
   Na het samenvoegen van de verschillende bronnen, moeten we de relatie tussen de bron- en doelrijen definiëren. Elke detaillijn van een verkoopfactuur uit de tabel IHISTO zal een rij vormen in de ster. Dit type relatie wordt ook wel *One-to-One Mapping* genoemd in ETL4ALL. De tabel IHISTO bevat echter niet alleen detaillijnen uit verkoopfacturen, maar ook uit bestelbonnen, zendnota's, enz. Daarnaast zijn niet alle detaillijnen van een verkoopfactuur gerelateerd aan een verkoop (bijvoorbeeld commentaarlijnen). Dit maakt het noodzakelijk om een aantal voorwaarden te stellen waaraan een rij uit de tabel IHISTO moet voldoen, vooraleer een doelrij zal worden gecreëerd.

   Eerst wordt gecontroleerd of we daadwerkelijk te maken hebben met een detaillijn uit een verkoopfactuur. In Lijn BOB wordt een afzonderlijk dagboek aangemaakt voor elk type document. In deze gevalstudie werd het dagboek, genaamd *VER*, aangemaakt voor de verkoopfacturen. We testen bijgevolg voor elke rij uit de tabel IHISTO of het attribuut *DBK* (dagboek) de waarde *VER* heeft. Het tweede criterium waaraan elke rij moet voldoen is dat het attribuut *QTYDELIV* (het aantal verkochte eenheden) een waarde heeft groter dan nul én dat de artikelreferentie *REF* niet leeg is. Op die manier worden lijnen die niet gerelateerd zijn aan een verkoop verwijderd. Een rij die aan beide voorwaarden beantwoordt, wordt opgenomen als een doelrij. ETL4ALL voorziet ook hier voorgedefinieerde functies voor het uitvoeren van *One-to-One Mapping* en het testen van de voorwaarden.

   Na het uitvoeren van de procedures, hebben we echter ondervonden dat een aantal klanten ontbreken in de tabel COMPAN waarvoor toch facturen bestaan. Dit zijn fouten in de database en daarom beslissen we de overeenkomstige facturen niet op

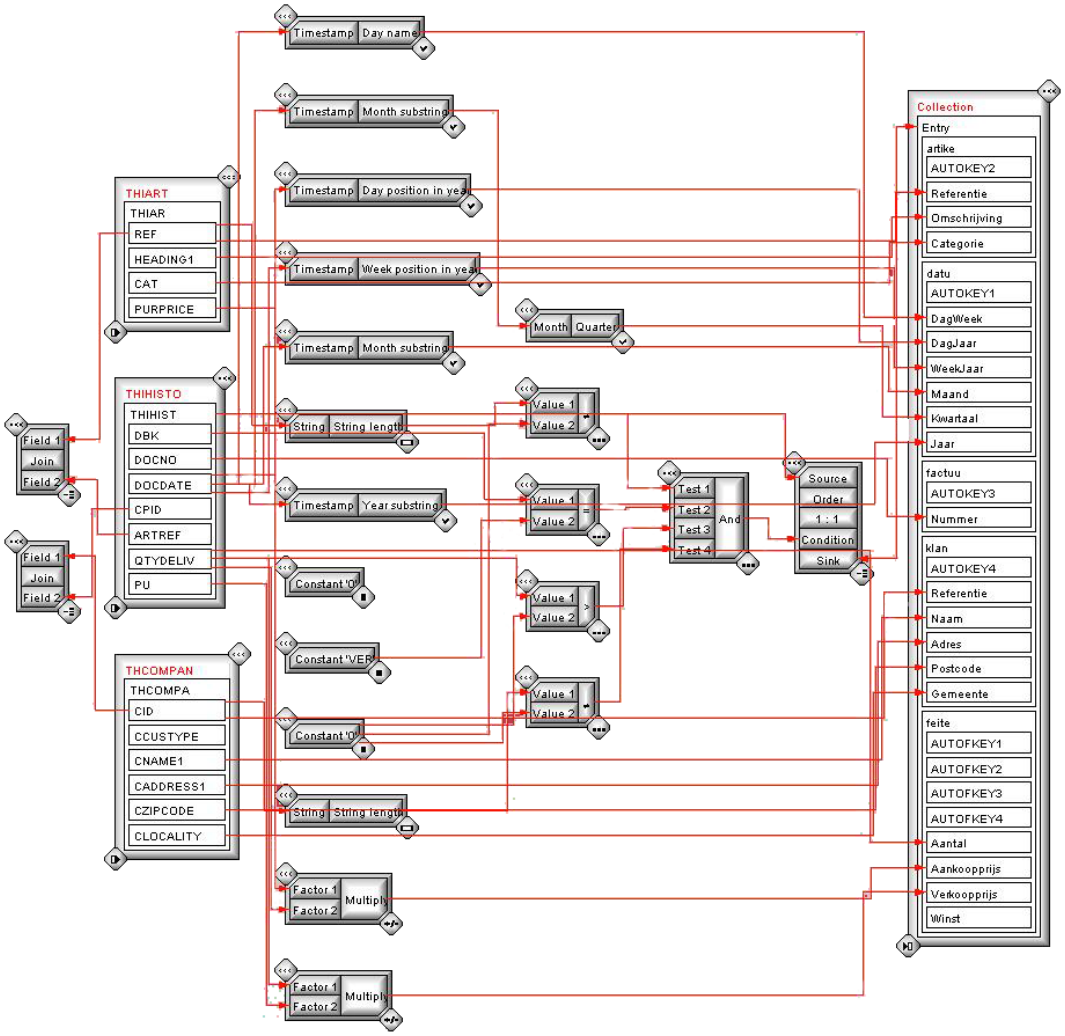
te nemen in de datamart. We voegen een derde criterium toe, namelijk dat de klantenreferentie na het samenvoegen van de tabellen niet leeg mag zijn. Hiervoor gaan we na of de lengte van de waarde in het veld *CID* (COMPAN) groter is dan nul.

3. *Het overnemen van ongewijzigde attributen*
   Voor elk rij uit de tabel IHISTO die voldoet aan de gestelde voorwaarden, wordt een doelrij aangemaakt in de ster. Voor elke doelrij moet vervolgens worden aangegeven hoe de waarden voor de attributen in de ster worden berekend. Zo zijn er een aantal waarden die rechtstreeks kunnen worden overgenomen uit de tabellen IART, IHISTO en COMPAN. Deze waarden zijn: artikelreferentie (*REF*), artikelomschrijving (*HEADING1*), categorie (*CAT*), factuurnummer (*DOCNO*), klantreferentie (*CID*), naam (*CNAME1*), adres (*CADDRESS1*), postcode (*CZIPCODE*), gemeente (*CLOCALITY*) en aantal verkochte eenheden (*QTYDELIV*).
4. *Het transformeren van de factuurdatum*
   De factuurdatum moet worden opgesplitst in verschillende componenten. ETL4ALL beschikt over een uitgebreid aanbod functies die het omzetten van een datum erg eenvoudig maken. De attributen *DagWeek*, *DagJaar*, *WeekJaar*, *Maand*, *Kwartaal* en *Jaar* worden op die manier snel berekend.
5. *De berekening van de aankoop- en verkoopprijs*
   De aankoopprijs wordt berekend aan de hand van volgende eenvoudige formule: *PURPRICE* (IART) * *QTYDELIV* (IHISTO). De verkoopprijs wordt analoog berekend namelijk: *PU* (IHISTO) * *QTYDELIV* (IHISTO).

- ***berekenWinst*** (figuur 7.10)
  In deze procedure wordt slechts één transformatie uitgevoerd, namelijk de winstberekening. Voor deze berekening nemen we het verschil tussen de verkoopprijs en de aankoopprijs. Zowel de bron als het doel van deze transformatie is de metadata definitie *ster*. De overige velden van het sterschema worden ongewijzigd overgenomen.

7.3.4 Stap 3: Het samenstellen van een programma

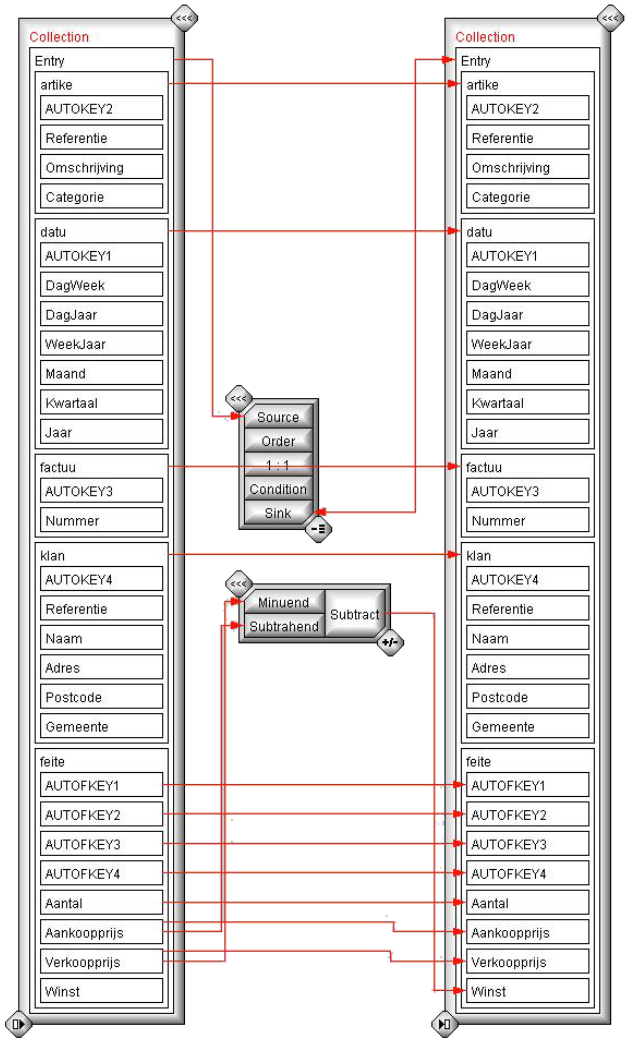
Met behulp van de procedures uit stap twee, maken we een eenvoudig programma: *maakDatamart*. We voeren eerst de procedure *vulDatamart* uit en nadien de procedure *berekenWinst* (figuur 7.11).



Figuur 7.9: De procedure *vulDatamart*

7.3.5 Stap 4: Het uitvoeren van het programma via een virtuele gegevensbron

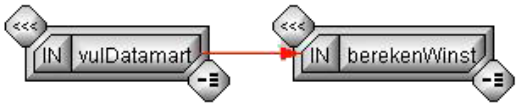
We creëren een virtuele gegevensbron die we *datamart* noemen. We selecteren daarbij het programma *maakDatamart* en de tabellen IART, IHISTO en COMPAN als input gegevensbronnen (figuur 7.12). Vervolgens inspecteren we de gegevens in de virtuele gegevensbron na het uitvoeren van het programma. De resulterende tabel geeft de verwachte resultaten.



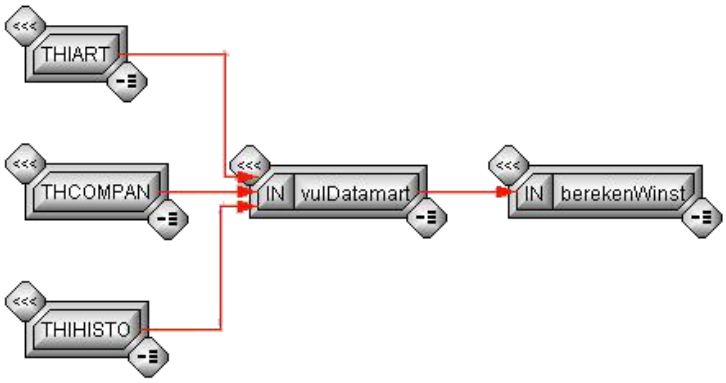
Figuur 7.10: De procedure *berekenWinst*

7.3.6 Stap 5: De resultaten naar een databasetabel sturen

In de laatste stap worden de gegevens uit de virtuele gegevensbron *datamart* fysiek opgeslagen in de databasetabellen van de datamart. Deze reële gegevensbron wordt in ETL4ALL aangeduid met de naam *ster*. Bij het opslaan van de resultaten, genereert ETL4ALL automatisch de primaire sleutels van de dimensietabellen (Artikel, Datum, Factuur en Klant) als ook de samengestelde sleutel van de feitentabel (Feiten).



Figuur 7.11: Het programma *maakDatamart*



Figuur 7.12: De virtuele gegevens van *datamart*

7.4 Samenstelling van een datamining tabel op basis van de datamart

7.4.1 Inleiding

Gegevens vormen de ruggengraat van datamining en kennisontsluiting. In de praktijk zijn bedrijfsgegegevens echter zelden of nooit beschikbaar in een geschikte vorm voor datamining. De grootste uitdaging voor dataminers bestaat daarom uit het voorbereiden van gegevens voor modellering. Om deze taken efficiënt uit te voeren, kan worden beroep gedaan op datawarehousetechnologie. Deze strategie passen we ook toe bij de uitwerking van onze gevalstudie. In voorgaande paragraaf werden de gegevens uit de Paradoxdatabase van Lijn BOB getransformeerd, opgeschoond en in een datamart geladen. In deze paragraaf extraheren we geschikte gegevens voor datamining uit de datamart. De ideale gegevensstructuur voor datamining bestaat uit rijen (observaties) en kolommen (variabelen) waarbij de rijen informatie bevatten over individuele cases (bijvoorbeeld klanten of producten) en de kolommen de attributen van de individuele gevallen beschrijven.

7.4.2 Observaties en variabelen voor datamining

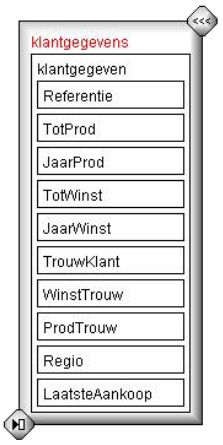
Door toepassing van datamining trachten we een aantal interessante verbanden te ontdekken tussen een aantal eigenschappen van de klanten. Op basis van de gegevens in de datamart, wordt een tabel samengesteld waarin elke rij overeenstemt met een individuele klant. Elke klant wordt vervolgens beschreven aan de hand van onderstaande attributen:

- ***TotProd***: Het totaal aantal producten dat een klant heeft gekocht.
- ***JaarProd***: Het aantal producten dat een klant kocht gedurende de laatste 12 maanden (referentiedatum = 02/2004).
- ***TotWinst***: De totale winst die een klant opleverde.
- ***JaarWinst***: De winst die een klant opleverde gedurende de laatste 12 maanden (referentiedatum = 02/2004).
- ***TrouwKlant***: Het aantal maanden tussen de eerste aankoop van een klant en de laatste aankoop.
- ***ProdTrouw***: Het gemiddeld aantal aankopen van een klant per maand (de ratio van TotProd en TrouwKlant).
- ***WinstTrouw***: De gemiddelde winst van een klant per maand (de ratio van TotWinst en TrouwKlant).
- ***Regio***: De postcode van een klant.
- ***LaatsteAankoop***: De laatste maal dat een klant een product aankocht, uitgedrukt in het aantal maanden ten opzichte van de referentiedatum 02/2004.

7.4.3 Transformaties voor het berekenen van de variabelen

Het uitvoeren van de transformaties voor het berekenen van de waarden van de variabelen, volgt een analoge werkwijze als het opvullen van de datamart. We beperken ons in deze paragraaf dan ook tot een beknopte bespreking van de vereiste transformaties. In bijlage E zijn de visuele voorstellingen opgenomen van de toegepaste procedures.

We gaan uit van een nieuwe metadata definitie die we *klantgegevens* noemen. Deze definitie bestaat uit tien velden, namelijk de klantreferentie en de negen variabelen die we wensen te berekenen voor elke klant (figuur 7.13). Deze metadata definitie bepaalt de doelgegevens van de uitgevoerde transformaties.



Figuur 7.13: De metadata definitie *klantgegevens*

De brongegevens zijn afkomstig uit de reële gegevensbron *ster*. Om de brongegevens te converteren naar de gewenste doelgegevens zijn volgende transformaties nodig:

1. De gegevens uit de datamart moeten worden gegroepeerd per klant.
2. De variabelen TotProd, JaarProd, TotWinst, JaarWinst, TrouwKlant en LaatsteAankoop moeten worden berekend per klant.
3. De ratio's ProdTrouw en WinstTrouw moeten worden bepaald op basis van de waarden voor TotProd, TotWinst en TrouwKlant.

De variabelen JaarProd, JaarWinst, TrouwKlant en LaatsteAankoop kunnen echter niet rechtstreeks worden berekend uit de gegevens in de datamart en vereisen daarom enkele tussenstappen.

7.4.3.1 Tussenstap 1: berekening van TrouwKlant en LaatsteAankoop

Voor de berekening van de variabelen TrouwKlant en LaatsteAankoop, moeten we achterhalen in welke maand en in welk jaar een klant zijn eerste aankoop deed, evenals de laatste maand en het laatste jaar waarin hij een product kocht. Aangezien de velden maand en jaar gescheiden zijn in de datamart, wordt deze berekening iets complexer. De eerste stap bestaat uit het combineren van de velden maand en jaar in één enkele waarde. We creëren hiervoor een nieuwe metadata definitie *berekenWaarde*. Deze definitie bestaat enerzijds uit de klantenreferentie en anderzijds uit de nieuwe waarde die is samengesteld uit de maand en het jaar. Om deze gegevens te berekenen, maken we een procedure *berekenWaarde* aan (figuur E.1). Met behulp van deze procedure wordt voor elk rij uit de datamart, een doelrij geplaatst in de tabel *berekenWaarde*. Van elk bronrij wordt de klantenreferentie overgenomen in de doelrij. De nieuwe waarde wordt

daarna als volgt berekend: $(JAAR * 100) + MAAND$. De procedure wordt uitgevoerd door het programma *vulWaarde*. Tenslotte wordt de virtuele gegevensbron *BerekenWaarde* gecreëerd die het programma *vulWaarde* uitvoert met als input, de reële gegevensbron *ster*.

In een volgende stap wordt de minimum- en maximumwaarde berekend per klant. We maken een nieuwe metadata definitie aan, *klantentrouw*, die bestaat uit de volgende drie velden: klantreferentie, minimumwaarde en maximumwaarde. De procedure *berekenTrouwKlant* groepeert de rijen uit de tabel *BerekenWaarde* per referentie (*N-to-one mapping*), en berekent vervolgens de minimum- en maximumwaarde van elke klant (figuur E.2). Via de virtuele gegevensbron *Klantentrouw* wordt het programma *vulTrouwKlant* uitgevoerd, dat de procedure *berekenTrouwKlant* bevat.

Tenslotte moeten de minimum- en maximumwaarde opnieuw worden vertaald naar een maand en een jaar. Dit gebeurt met behulp van de metadata definitie *klantentrouwVervolg*. De eerste procedure *berekenTrouwKlantVervolg* berekent het eerste jaar en het laatste jaar door de minimum- en maximumwaarde uit de tabel *klantentrouw* te delen door 100 en naar beneden af te ronden (figuur E.3). De tweede procedure *berekenTrouwKlantVervolg2* berekent dan de eerste maand van het eerste jaar en de laatste maand van het laatste jaar als volgt: De tabellen *klantentrouwVervolg* en *klantentrouw* worden samengevoegd en vervolgens worden het eerste jaar en het laatste jaar uit de tabel *klantentrouwVervolg* vermenigvuldigd met 100. Nadien worden deze waarden afgetrokken van de oorspronkelijke mininum- en maximumwaarde in de tabel *klantentrouw*. Zo worden de eerste en laatste maand bekomen (figuur E.4).

7.4.3.2 Tussenstap 2: berekening van JaarProd en JaarWinst

Voor het berekenen van het aantal gekochte producten en de winst gedurende de laatste 12 maanden (referentiedatum = 02/2004), wordt de metadata definitie *prodWinJaar* gecreëerd. Deze bestaat uit de klantreferentie, het aantal gekochte producten en de winst. Vervolgens worden door middel van twee procedures de gevraagde waarden berekend. De eerste procedure *berekenProdWinJaar* selecteert de rijen uit de datamart die tot de periode van maart 2003 tot en met februari 2004 behoren (figuur E.5). De procedure *berekenSom* groepeert deze rijen per klant en sommeert het aantal en de winst (figuur E.6). Het programma *vulProdWinJaar* voert achtereenvolgens de procedures *berekenProdWinJaar* en *berekenSom* uit, en wordt weergegeven door middel van de virtuele gegevensbron *ProdWinJaar*.

7.4.3.3 Berekening van de klantgegevens

Na het uitvoeren van de tussenstappen, kan de tabel *klantgegevens* vrij eenvoudig worden opgevuld. De procedure *vulKlantgegevens* voert volgende taken uit (figuur 7.14):

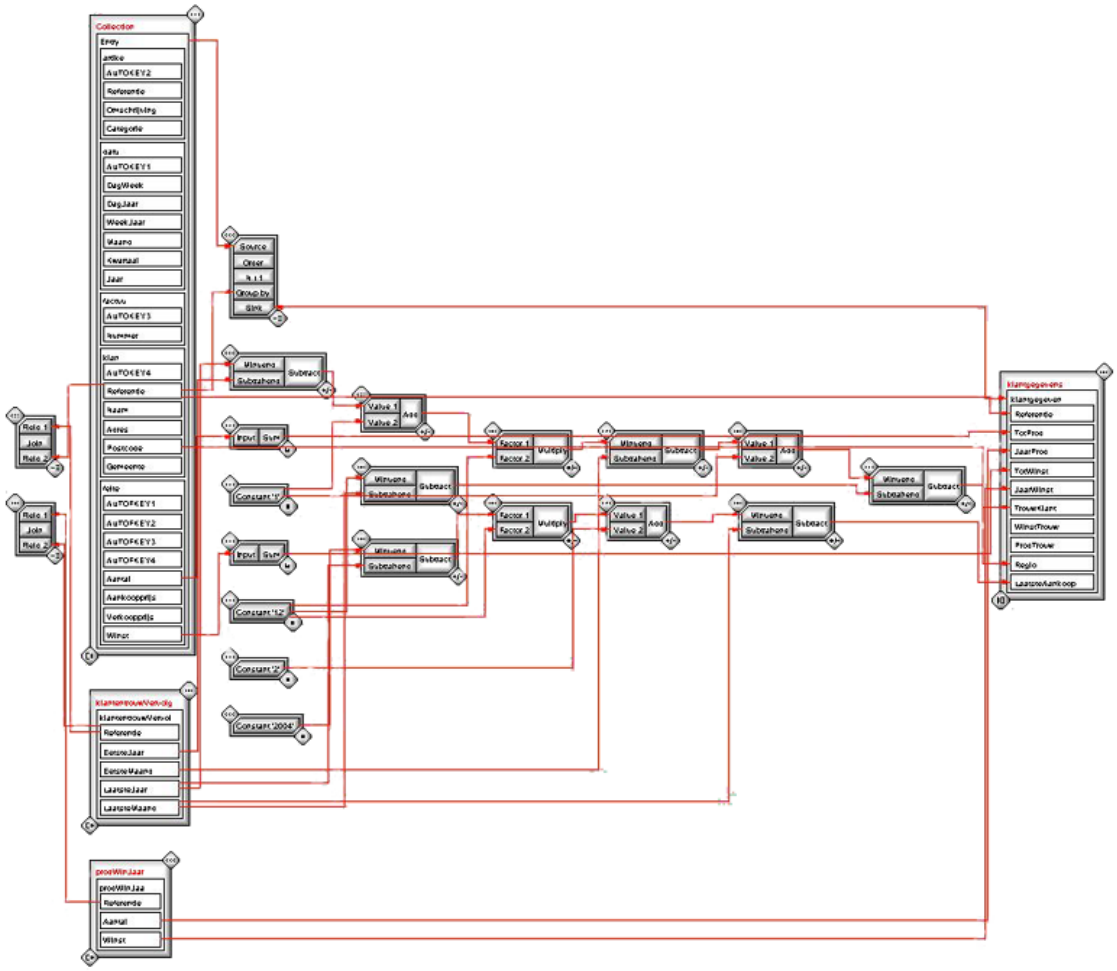
- De metadata definities *ster*, *klantentrouwVervolg* en *prodWinJaar* worden samengevoegd en de rijen worden gegroepeerd per klantreferentie.
- De waarden voor de referentie en de regio worden rechtstreeks overgehaald uit de metadata definitie *ster*.
- De waarden TotProd en TotWinst worden berekend door het sommeren van de velden *Aantal* en *Winst* (*ster*).
- De waarden JaarProd en JaarWinst worden zonder extra berekening overgenomen uit de metadata definitie *prodWinJaar*
- De variabele TrouwKlant wordt bepaald met behulp van volgende formule: (*LaatsteJaar* - *EersteJaar*+ 1)* 12)- (*EersteMaand* + 1)-(12 - *LaatsteMaand*)
- De variabele LaatsteAankoop wordt berekend met de formule: ((2004 - *LaatsteJaar*) * 12) + 2 - *LaatsteMaand*

In de procedure *berekenRatios* worden de ratio's WinstTrouw en ProdTrouw berekend. Bovendien wordt nagegaan welke klanten geen aankoop hebben verricht gedurende de laatste 12 maanden. Aan deze klanten wordt de waarde 0 toegekend voor de variabelen *JaarProd* en *JaarWinst* (figuur 7.15).

Het programma *vulKlantgegevens* voert de twee bovenstaande procedures na elkaar uit. Tenslotte worden de resultaten bekomen door het uitvoeren van het programma via de virtuele gegevensbron *Klantgegevens*. We wensen de resultaten echter ook fysiek op te slaan om zo de gegevens in een datamining toepassing te kunnen gebruiken. We kopiëren daarom de gegevens uit de gegevensbron *Klantgegevens* naar een databasetabel.

7.5 Conclusie na het gebruik van een ETL-toepassing

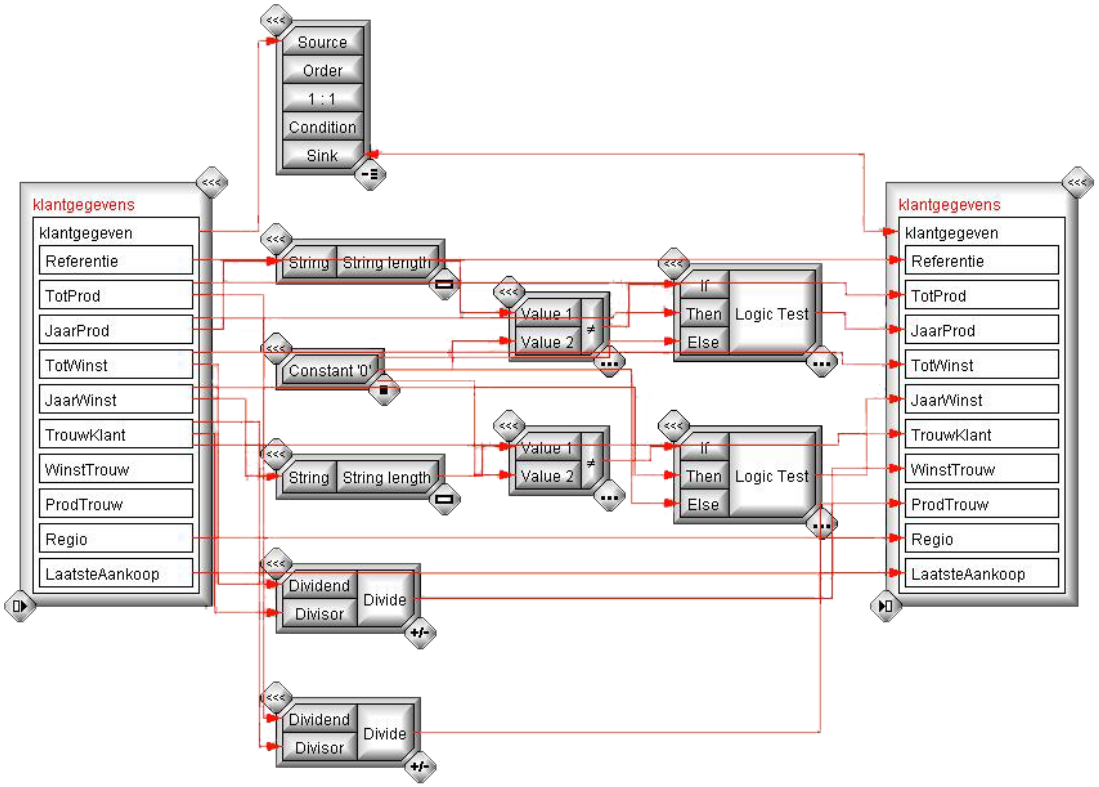
Het construeren en opvullen van de datamart bleek een vrij eenvoudige taak te zijn. Dit kan enerzijds worden verklaard door het eenvoudige datamartontwerp, maar anderzijds ook door het gebruik van slechts één gegevensbron. Hierdoor kwam één van de voornaamste eigenschappen van datawarehousing, namelijk het integreren van gegevens uit diverse bronnen, niet duidelijk naar voren. Verder biedt de grafische interface van ETL4ALL een unieke en overzichtelijke weergave van de metadata definities, de procedures en de programma's. Het aanbod voorgedefinieerde functies was voldoende en maakte het samenstellen van transformatieprocedures eenvoudiger. Tijdens het transformatieproces werd slechts één probleem vastgesteld, namelijk het vinden van een geschikt stuurprogramma (*driver*) voor het beheren van de communicatie



Figuur 7.14: De procedure *vulKlantGegevens*

tussen de Paradoxtabellen en ETL4ALL.

Door het construeren van een datamart op basis van de gegevens in Lijn BOB, werden twee doelstellingen bereikt. Enerzijds werd een overzichtelijke, opgeschoonde en betrouwbare database gecreëerd voor het uitvoeren van een analyse. Anderzijds werd een databasestructuur bekomen die het opvragen van informatie vereenvoudigt. Dit werd positief ervaren tijdens het berekenen van de datamining variabelen. Er werd enkel hinder ondervonden door het omrekenen van datums en de vrij omslachtige werkwijze die het gevolg is van de opbouw van transformaties uit metadata definities, procedures, programma's en virtuele gegevensbronnen.



Figuur 7.15: De procedure *berekenRatios*

Hoofdstuk 8

Ontwikkeling van een datamining model in PMML met behulp van SAS Enterprise Miner

In dit hoofdstuk worden de verschillende fasen doorlopen in het ontwikkelingsproces van een datamining model. Hierbij wordt gebruik gemaakt van een datamining toepassing. Na een korte inleiding, worden de verschillende stappen concreet toegepast voor de gevalstudie. Tenslotte wordt het resulterende model vertaald in PMML.

8.1 Inleiding

Voor het toepassen van datamining bestaan diverse softwarepakketten die het datamining proces automatiseren. Uit dit ruime aanbod, kozen we SAS Enterprise Miner 5.1. Deze applicatie stroomlijnt het volledige datamining proces vanaf de gegevenstoegang tot het evalueren van modellen, door alle noodzakelijke taken te ondersteunen in één geïntegreerd systeem (SAS 2005). We kozen voor Enterprise Miner omdat deze de mogelijkheid voorziet om modellen in PMML te genereren.

Het doel van dit hoofdstuk is het bekomen van een zinvol model op basis van de afgeleide variabelen uit de datamart en dit model vervolgens te vertalen naar PMML code. Meer bepaald gaan we op zoek naar verbanden tussen de winst gedurende het laatste jaar (JaarWinst) en de overige variabelen. Op basis van deze relaties hopen we inzicht te bekomen in de eigenschappen van de klant die bepalend zijn voor de winst van het afgelopen jaar.

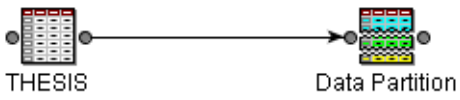
8.2 SEMMA

Het acroniem SEMMA - sample, explore, modify, model, assess - verwijst naar de kernprocessen van datamining. SEMMA is geen datamining methodologie maar eerder een logische organisatie van de functionele instrumenten van SAS Enterprise Miner voor het uitvoeren van de belangrijkste taken van datamining. SEMMA is voornamelijk gericht op de modelontwikkeling binnen het datamining proces (SAS 2005) :

- **Sample:** Het identificeren van de input datasets (het identificeren van de brongegevens, het nemen van een steekproef van een grote gegevensverzameling, het opsplitsen van data in training, validatie en test datasets).
- **Explore:** Statistische en grafische verkenning van de gegevens (het uitzetten van de data, het bekomen van beschrijvende statistieken, het identificeren van belangrijke variabelen).
- **Modify:** De gegevens voorbereiden voor analyse (het creëren van additionele variabelen, het transformeren van bestaande variabelen, het identificeren van uitschieters, het imputeren van ontbrekende waarden, de manier aanpassen waarop de variabelen worden gebruikt voor analyse).
- **Model:** Het bouwen van een model (het modelleren van een doelvariabele door gebruik te maken van bijvoorbeeld een regressiemodel, een beslissingsboom of een neuraal netwerk).
- **Assess:** Het vergelijken en evalueren van verschillende modellen.

8.2.1 Sample: Identificatie van de input datasets

In de eerste fase van het SEMMA proces, maken we gebruik van twee instrumenten in Enterprise Miner, namelijk de *Input Data Source* node (genaamd 'THESIS') en de *Data Partition* node (figuur 8.1).



Figuur 8.1: Sample

Met behulp van de *Input Data Source* node, worden de gegevensbronnen ingelezen en de variabelen geïdentificeerd. De brontabel, die werd afgeleid in voorgaand hoofdstuk, bevat 835 observaties en 10 variabelen. De voornaamste eigenschappen van de variabelen zijn het meetniveau en de rol die de variabele zal spelen in het model. We identificeren de variabele JaarWinst

als doelvariabele (*target*). De variabele Klantreferentie zal de rol van ID vervullen aangezien deze waarde uniek is voor elke klant. De overige variabelen krijgen voorlopig de rol van input variabele. De meeste variabelen bevatten continue waarden en worden daarom aangeduid met *interval*. Uitzonderingen zijn de variabele Klantreferentie en Regio. Deze bevatten discrete waarden zonder een logische volgorde en zijn daarom *nominaal*. Tabel 8.1 geeft een overzicht van de variabelen en hun kenmerken.

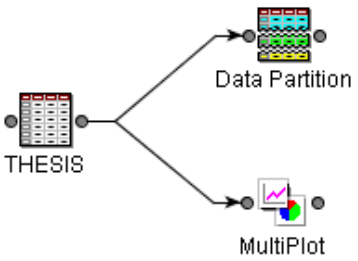
Tabel 8.1: De variabelen en hun eigenschappen

Variabele	**Modelrol**	**Meetniveau**
JaarProd	Input	Interval
JaarWinst	Target	Interval
Klantreferentie	ID	Nominaal
LaatsteAankoop	Input	Interval
ProdTrouw	Input	Interval
Regio	Input	Nominaal
TotProd	Input	Interval
TotWinst	Input	Interval
TrouwKlant	Input	Interval
WinstTrouw	Input	Interval

Aangezien de gegevensbron slechts 835 observaties bevat, is het niet nodig een steekproef te nemen uit deze tabel. De *Data Partition* node laat ons vervolgens toe de dataset op te splitsen in een training, validatie en test dataset. De training dataset wordt gebruikt voor het bouwen van het model, de validatie dataset wordt gebruikt voor het controleren en evalueren van het model en de test dataset is een additionele dataset die kan worden gebruikt voor modelevaluatie. De opsplitsing gebeurt op basis van percentages. In onze gevalstudie behoort 60 procent van de gegevens tot de training data en 40 procent tot de validatie data. Er worden geen testgegevens voorzien. Deze keuze werd gemaakt omwille van de beperkte grootte van de dataset.

8.2.2 Explore: Verkennende gegevensanalyse

Voor de verkennende gegevensanalyse, maken we gebruik van de *Multiplot* node (figuur 8.2). Met behulp van de *Multiplot* node kan men grote hoeveelheden gegevens grafisch verkennen, de verdelingen van de gegevens observeren en de relaties tussen variabelen onderzoeken. *Multiplot* geeft extreme waarden weer en helpt bij het ontdekken van patronen en trends. Bovendien worden van elke variabele een aantal beschrijvende statistieken berekend. Bijlage F bevat de grafische voorstellingen (frequenties) van de continue variabelen.



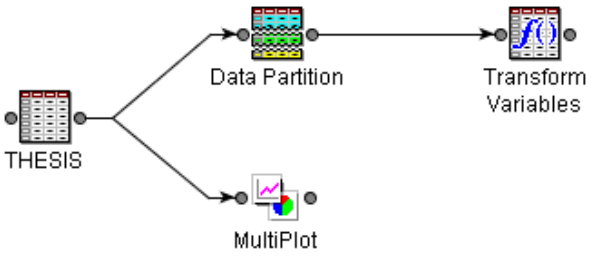
Figuur 8.2: Explore

De resultaten van de *Multiplot* node geven aan dat er geen ontbrekende waarden zijn. Dit is het gevolg van de transformaties in ETL4ALL. We stellen echter wel vast dat een aantal variabelen, voornamelijk JaarProd, ProdTrouw en TotProd, een assymetrische verdeling hebben. Dit wordt niet alleen weergegeven door de grafieken, maar ook door de hoge waarden voor de *skewness* of scheefheid. In zeer assymetrische verdelingen, kan een klein percentage van de waarden een grote invloed uitoefenen. Door het toepassen van transformaties in de volgende stap van het SEMMA proces, trachten we deze input variabelen te transformeren naar een meer normale verdeling zodanig dat we een beter model zullen bekomen.

We stellen ook vast dat er een evenredigheid bestaat tussen de variabele JaarProd en JaarWinst. Deze onderlinge relatie is voor de hand liggend aangezien een toename in het aantal verkochte producten ook een winststijging met zich meebrengt. Aangezien we echter de winst gedurende het laatste jaar trachten af te leiden uit de overige variabelen, en we veronderstellen dat het aantal verkochte producten daarbij een onbekend gegeven is, verwerpen we de variabele JaarProd uit het model. Op die manier vermijden we dat het model de doelvariabele enkel en alleen zou voorspellen op basis van het aantal verkochte eenheden gedurende het laatste jaar.

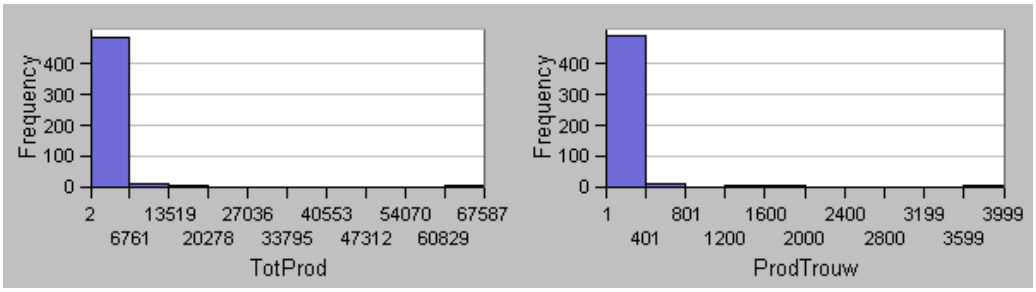
8.2.3 Modify: Transformatie van de variabelen

We maken gebruik van de *Transform Variables* node om de variabelen met een assymetrische verdeling te transformeren (figuur 8.3).



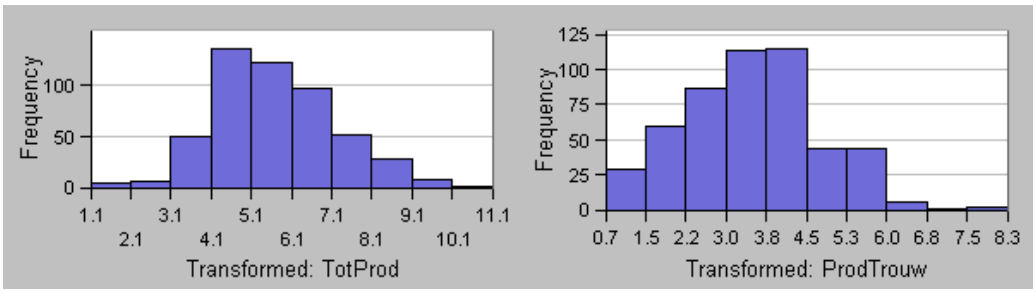
Figuur 8.3: Modify

Mits de variabelen TotProd en ProdTrouw een scheefheid hebben van meer dan 10.0, transformeren we deze met behulp van de *Transform Variables* node. Deze variabelen bevatten steeds positieve waarden en bijgevolg kunnen we deze transformeren door het nemen van het logaritme. Het effect hiervan wordt geïllustreerd aan de hand van onderstaande histogrammen. Vóór het toepassen van het logaritme hadden de variabelen TotProd en ProdTrouw een scheefheid van 13.1678 respectievelijk 12.7009 (figuur 8.4).



Figuur 8.4: De oorspronkelijke verdeling van de variabele TotProd en ProdTrouw

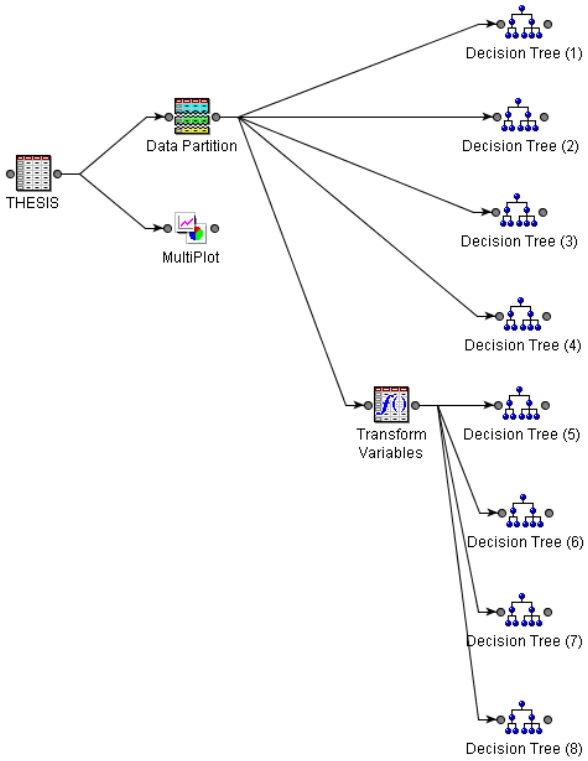
Na de transformatie, heeft de variabele TotProd een scheefheid van slechts 0.30357 en de variabele ProdTrouw een waarde van 0.19881 (figuur 8.5).



Figuur 8.5: De verdeling van het logaritme van de variabele TotProd en ProdTrouw

8.2.4 Model: Ontwikkeling van modellen

De voorlaatste stap in het SEMMA proces, bestaat uit het effectief ontwikkelen van enkele datamining modellen. Hierbij maken we gebruik van beslissingsbomen omdat deze modellen zeer goed interpreteerbaar zijn (figuur 8.6).



Figuur 8.6: Model

Met behulp van de *Decision Tree* node, bouwen we een aantal beslissingsbomen. Elke boom bezit volgende kenmerken:

- *Splitsingscriterium*: Specificeert de methode voor het zoeken naar en het evalueren van mogelijke kandidaten voor splitsings- of vertakkingsregels. De keuze voor de geschikte methode is afhankelijk van de doelvariabele. Voor een continue doelvariabele bestaan twee methoden namelijk het reduceren van de variantie en de p-waarde[1] van de F-test[2] die geassocieerd is met de variantie in een knop. We passen beide methoden toe.

- *Significantieniveau*: Specificeert de drempel voor de p-waarde van een kandidaat splitsingsregel. Deze waarde houden we constant op 0.20.

[1] Significantie
[2] Variantie-analyse

- *Ontbrekende waarden*: Geeft aan hoe een splitsingsregel een observatie behandelt met ontbrekende waarden. Aangezien er geen ontbrekende waarden zijn in de brongegevens, behouden we hiervoor de standaardwaarde die aangeeft dat ontbrekende waarden zullen worden gebruikt bij het zoeken naar splitsingsregels.

- *Bladgrootte*: Het kleinste aantal training observaties dat een blad mag hebben. Deze waarde wordt vastgelegd op vijf observaties.

- *Maximum aantal vertakkingen*: Beperkt het aantal vertakkingen dat een splitsingsregel kan produceren. We passen zowel een splitsing van twee vertakkingen toe (binair) als een splitsing van vier vertakkingen.

- *Maximale diepte*: Specificeert het maximaal aantal generaties van knopen. Dit getal wordt gelijkgesteld aan zes.

- *Minimale categorische grootte*: Geeft het minimaal aantal observaties aan van een categorie. De categorie moet minstens voorkomen met dit aantal om te worden geselecteerd. We houden een waarde van vijf aan.

- *Aantal regels*: Specificeert hoeveel splitsingsregels worden bewaard in elke knoop. De beslissingsboom maakt slechts gebruik van één regel. De overige regels worden bewaard ter vergelijking. Tijdens het bouwen van de boom worden in elke knoop een aantal regels geëvalueerd door middel van het berekenen van een statistisch cijfer dat de sterkte aangeeft van elke regel. We leggen het aantal splitsingsregels vast op twee.

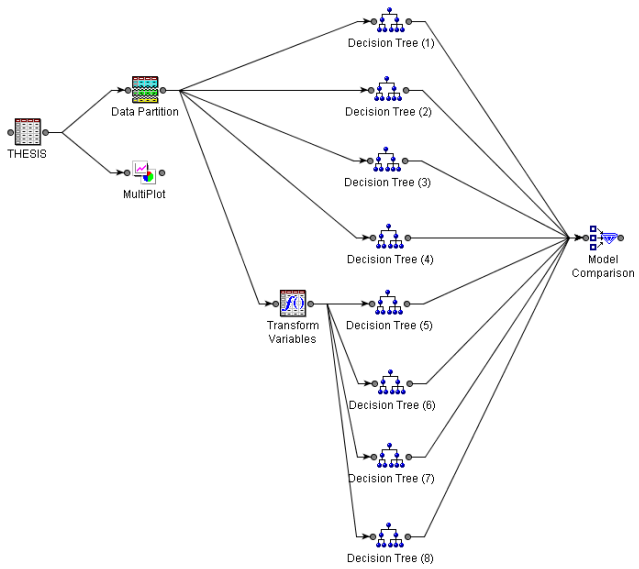
- *Aantal surrogaatregels*: Duidt het aantal surrogaatregels aan die zullen worden gezocht voor elke knoop (maar niet voor bladeren). Deze surrogaatregels zullen als backup fungeren voor de hoofdregel in het geval van ontbrekende waarden. Aangezien er geen ontbrekende waarden zijn, leggen we dit aantal vast op nul.

- *Splitsingsgrootte*: Geeft het kleinste aantal training observaties weer dat een knoop moet hebben om een split te overwegen. Deze waarde stellen we gelijk aan tien.

We bouwen vier beslissingsbomen met behulp van de oorspronkelijke variabelen en vier bomen met behulp van de getransformeerde variabelen. We laten daarbij het splitsingscriterium en het maximaal aantal vertakkingen variëren. De overige kenmerken behouden een constante waarde.

8.2.5 Assess: Evaluatie van de modellen

Tenslotte worden de ontwikkelde modellen vergeleken en geëvalueerd door middel van de *Model Comparison* node (figuur 8.7).



Figuur 8.7: Assess

Na het uitvoeren van de vergelijking, geven de resultaten aan dat boom vijf en boom acht de beste modellen zijn. Hoewel alle modellen vrij dicht bij mekaar liggen, scoren deze modellen beter op vlak van de *Averaged Squared Error* (tabel 8.2). Enterprise Miner geeft aan dat beslissingsboom vijf de beste keuze is. Dit is een binaire beslissingsboom met als splitsingscriterium het reduceren van de variantie.

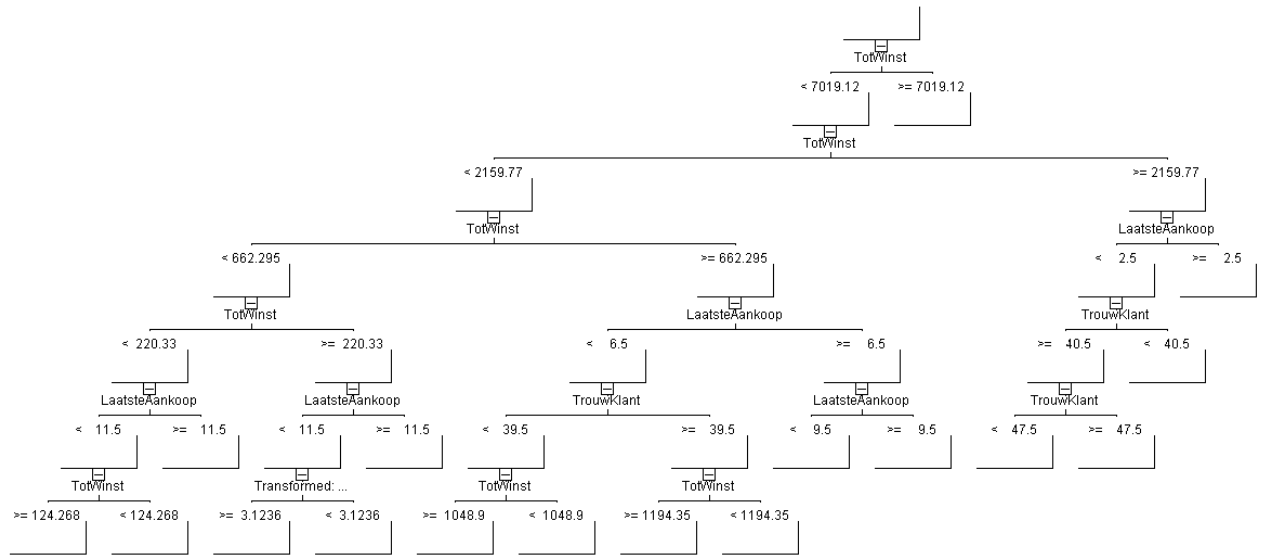
Tabel 8.2: Vergelijking van de modellen op basis van de *Average Squared Error (ASE)*

Model	**Transform.**	**Criterium**	**Aant. takken**	**ASE (training)**	**ASE (validatie)**
Boom (1)	nee	variantie	2	74006,77	149694,03
Boom (2)	nee	variantie	4	74006,77	149694,03
Boom (3)	nee	F-test	2	75700,59	149819,92
Boom (4)	nee	F-test	4	75741,60	151725,40
Boom (5)	ja	variantie	2	72764,70	151632,50
Boom (6)	ja	variantie	4	75700,59	149819,92
Boom (7)	ja	F-test	2	75741,60	151725,40
Boom (8)	ja	F-test	4	72764,70	151632,50

8.3 De beslissingsboom van de variabele JaarWinst

De resulterende beslissingsboom geeft een indicatie van de variabelen die invloed uitoefenen op de hoeveelheid winst per klant gedurende de afgelopen twaalf maanden (figuur 8.8). Bijlage G bevat een uitgebreide voorstelling van beslissingsboom vijf. De voornaamste kenmerken

zijn de totale hoeveelheid winst (TotWinst) en het aantal maanden sinds de laatste aankoop (LaatsteAankoop). Naarmate de totale hoeveelheid winst toeneemt, zal ook de winst van het laatste jaar toenemen. We merken echter op dat de variabele TotWinst ook de winst van het afgelopen jaar omvat. Voor de variabele LaatsteAankoop stellen we vast dat een klant die recenter heeft gekocht, meer winst zal opleveren dan een klant die minder recent heeft gekocht. Daarnaast bestaan er relaties tussen de variabele JaarWinst en de variabelen TrouwKlant, ProdTrouw en WinstTrouw.



Figuur 8.8: Visuele voorstelling van de beslissingsboom

8.4 Omzetting van het model naar PMML

Met behulp van SAS Enterprise Miner 5.1, kunnen datamining modellen vrij eenvoudig vertaald worden in PMML code. We overlopen de voornaamste onderdelen van het PMML document van de resulterende beslissingsboom en gaan na hoe dit model wordt voorgesteld door middel van PMML. Het volledige PMML document is opgenomen in bijlage H.

De header of hoofding van het PMML document wordt als volgt voorgesteld:

```
<Header copyright="Copyright(c) 2002 SAS Institute Inc.,
 Cary, NC, USA. All Rights Reserved.">
 <Application name="SAS(r)" version="9.1"/>
 <Timestamp>2005-05-07 17:19:05</Timestamp>
</Header>
```

Deze hoofding geeft het copyright, de versie van SAS die werd gebruikt en het tijdstip waarop het model werd ontwikkeld. Het volgende onderdeel is het gegevenswoordenboek:

```
<DataDictionary numberOfFields="8">
 <DataField name="JaarWinst" x-SASlabel="JaarWinst" optype="continuous"
 dataType="double"/>
 <DataField name="LOG_ProdTrouw" x-SASlabel="Transformed: ProdTrouw"
 optype="continuous" dataType="double"/>
 <DataField name="LOG_TotProd" x-SASlabel="Transformed: TotProd"
 optype="continuous" dataType="double"/>
 <DataField name="LaatsteAankoop" x-SASlabel="LaatsteAankoop"
 optype="continuous" dataType="double"/>
 <DataField name="TotWinst" x-SASlabel="TotWinst"
 optype="continuous" dataType="double"/>
 <DataField name="TrouwKlant" x-SASlabel="TrouwKlant"
 optype="continuous" dataType="double"/>
 <DataField name="WinstTrouw" x-SASlabel="WinstTrouw"
 optype="continuous" dataType="double"/>
 <DataField name="Regio" x-SASlabel="Regio"
 optype="categorical" dataType="string" x-SASformat="$4." x-SASinformat="$."/>
</DataDictionary>
```

Het gegevenswoordenboek definieert acht velden of variabelen. Dit zijn de oorspronkelijke variabelen met uitzondering van de klantreferentie die werd aangeduid als ID, de variabele JaarProd die werd geweigerd en de getransformeerde variabelen LOG_ProdTrouw en LOG_TotProd. De meeste variabelen zijn intervalvariabelen, enkel de variabele Regio is een categorische variabele.

Aan de hand van bovenstaande PMML code, stellen we vast dat de getransformeerde variabelen rechtstreeks worden opgenomen in het gegevenswoordenboek in plaats van de oorspronkelijke variabelen. Het door SAS gegenereerde PMML document definieert geen transformatiewoordenboek of locale transformaties. Hierdoor gaat belangrijke informatie betreffende de wijze van transformatie verloren.

Het voornaamste onderdeel van het PMML document is het model. In dit geval gaat het om een beslissingsboom die als volgt wordt ingeleid:

```
<TreeModel functionName="regression" splitCharacteristic="binarySplit">
```

Dit element definieert een beslissingsboom die wordt gebruikt voor regressiedoeleinden; de boom tracht namelijk een continue doelvariabele te voorspellen. Daarnaast wordt aangegeven

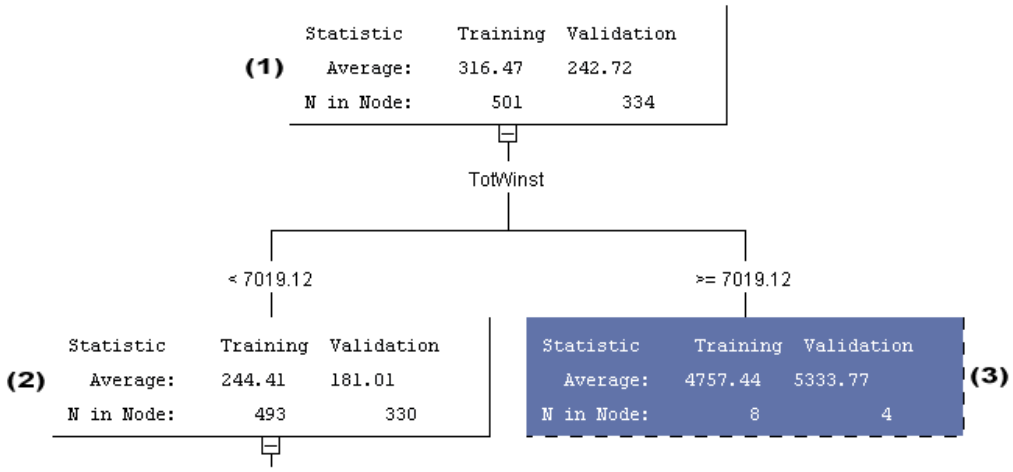
dat het aantal vertakkingen per knoop beperkt is tot twee. De eerstvolgende component in het PMML document, is het mining schema:

```
<MiningSchema>
 <MiningField name="JaarWinst" usageType="predicted" optype="continuous"/>
 <MiningField name="TotWinst" usageType="active" optype="continuous"/>
 <MiningField name="LaatsteAankoop" usageType="active" optype="continuous"/>
 <MiningField name="TrouwKlant" usageType="active" optype="continuous"/>
 <MiningField name="LOG_ProdTrouw" usageType="active" optype="continuous"/>
</MiningSchema>
```

Het mining schema definieert de variabelen die voorkomen in het model en de manier waarop ze worden gebruikt. We herkennen de inputvariabelen: TotWinst, LaatsteAankoop, TrouwKlant en LOG_ProdTrouw evenals de doelvariabele JaarWinst. Vervolgens worden de in totaal 33 knopen van de beslissingsboom beschreven. We beperken ons in deze tekst tot het overlopen van de eerste drie knopen. De PMML code van de overige knopen is vergelijkbaar met deze code.

```
<Node id="1" score="316.474367465" recordCount="501">
 <True/>
 <Node id="3" score="4757.4353" recordCount="8">
  <CompoundPredicate booleanOperator="surrogate">
   <SimplePredicate field="TotWinst" operator="greaterOrEqual" value="7019.12"/>
   <False/>
  </CompoundPredicate>
 </Node>
 <Node id="2" score="244.410092697" recordCount="493">
  <CompoundPredicate booleanOperator="surrogate">
   <SimplePredicate field="TotWinst" operator="lessThan" value="7019.12"/>
   <SimplePredicate field="TotWinst" operator="isMissing"/>
  </CompoundPredicate>
```

De eerste knoop die wordt beschreven, is de wortel. Deze krijgt als *id* het nummer één. SAS maakt gebruik van oplopende gehele getallen voor de nummering van de knopen. De nummering gebeurt van links naar rechts en van boven naar beneden per vertakkingsniveau (figuur 8.9).



Figuur 8.9: De eerste drie knopen van de beslissingsboom

Het attribuut *score* geeft de gemiddelde waarde weer voor de JaarWinst van de observaties in de knoop. Het attribuut *recordCount* geeft het aantal observaties in de training dataset die tot de knoop behoren. De wortel bevat alle aanwezige observaties en heeft bijgevolg als predikaat de waarde <*true/*>. De wortel wordt verder opgedeeld in twee vertakkingen. De volgorde waarin de knopen worden beschreven in PMML, is per tak en van links naar rechts, uitgezonderd indien de rechtse knoop niet verder vertakt. In het laatste geval wordt eerst de rechtse knoop beschreven. Toegepast op de code van de beslissingsboom uit de gevalstudie, betekent dit dat eerst knoop drie wordt beschreven. Ook voor deze knoop wordt de gemiddelde winst en het aantal observaties in de training set aangeduid. Daarnaast bevat dit element een samengesteld predikaat met als operator de waarde *surrogate*. Concreet betekent dit dat indien een observatie een waarde heeft voor de variabele TotWinst, het eerste predikaat wordt gecontroleerd namelijk: TotWinst >= 7019,12. Indien de waarde voldoet aan dit criterium, wordt de observatie opgenomen in deze knoop. Als de waarde van de variabele TotWinst ontbreekt, wordt het tweede predikaat uitgevoerd waardoor het resultaat de waarde *FALSE* aanneemt. De observatie behoort bijgevolg niet tot de knoop. De beschrijving van knoop twee is gelijkaardig aan die van knoop drie, maar hier geeft de surrogaatregel de waarde *TRUE* in geval van een ontbrekende waarde als gevolg van de operator *isMissing*. Deze manier van omgaan met ontbrekende waarden wordt in elke vertakking toegepast. Een observatie met een ontbrekende waarde voor het splitsingscriterium wordt daardoor steeds ondergebracht in de linkse vertakking.

Hoofdstuk 9

Conclusies

Dit hoofdstuk geeft een antwoord op de centrale onderzoeksvraag die werd omschreven in de probleemstelling. Op basis van de resultaten van de gevalstudie, wordt een algemene conclusie geformuleerd voor het gestelde probleem.

9.1 Beantwoording van de centrale onderzoeksvraag

In het hoofdstuk Probleemstelling werd het algemene praktijkprobleem omschreven van de tekortkomingen van ERP-systemen met betrekking tot het voorzien van managementinformatie. Een mogelijke oplossing werd aangereikt door het integreren van Business Intelligence toepassingen, die het onsluiten van kennis uit de ERP-gegevensbron mogelijk maken. In deze context situeerde zich de opzet van deze eindverhandeling. We onderzochten de uitvoerbaarheid van het integreren van een dataminig model in een ERP-systeem, door middel van de Predictive Model Markup Language (PMML). Deze doelstelling werd vertaald in volgende centrale onderzoeksvraag:

Kan een datamining model geïmplementeerd worden in een ERP-systeem met behulp van PMML?

Het antwoord op deze onderzoeksvraag wordt opgesplitst naar volgende deelvragen:

1. Op welke wijze kan een datamining model geïmplementeerd worden in een ERP-systeem door toepassing van PMML?
2. Is het integreren van een datamining model in een ERP-systeem door middel van PMML uitvoerbaar in de praktijk?

9.1.1 Op welke wijze kan een datamining model geïmplementeerd worden in een ERP-systeem door toepassing van PMML?

Uit de gevalstudie blijkt dat het rechtstreeks uitvoeren van een datamining analyse op een ERP-database niet optimaal is. Een ERP-database biedt doorgaans geen stabiele basisomgeving voor analyse aangezien gegevens continu worden gewijzigd. Daarnaast bevinden deze gegevens zich meestal niet in een geschikte vorm voor datamining. Een bijkomend probleem is de ongewenste belasting van het ERP-systeem tijdens het uitvoeren van een analyse. Deze beperkingen vloeien voort uit het feit dat een ERP-systeem een operationeel systeem is, ontworpen voor het vastleggen van transacties en niet voor het toepassen van datamining.

Bovenstaande beperkingen van een ERP-systeem kunnen worden opgevangen door het voorzien van een afzonderlijk databasestructuur die naast de bestaande ERP-database functioneert en is afgestemd op de behoeften van analysetoepassingen. Voor de gevalstudie betekent dit concreet het bouwen van een datamart. De structuur van de datamart wordt ontwikkeld aan de hand van een dimensionaal gegevensmodel. Met behulp van een ETL-toepassing worden de vereiste gegevens uit de ERP-database geëxtraheerd en omgevormd om uiteindelijk in de datamart te worden geladen. Op die manier creëert een datamart een stabiele omgeving met kwalitatief hoogwaardige gegevens, geschikt voor datamining, zonder daarbij het ERP-systeem onnodig te belasten.

Op basis van de gegevens in de datamart, kan een datamining tabel worden samengesteld bestaande uit waarnemingen en variabelen. Op deze gegevens kan vervolgens een datamining analyse worden uitgevoerd met behulp van een datamining toepassing. Na het vertalen van het resulterende model in PMML code, kan het worden geëxporteerd naar de ETL-toepassing waar het rechtstreeks wordt uitgevoerd op de gegevens afkomstig van het ERP-systeem.

9.1.2 Is het integreren van een datamining model in een ERP-systeem door middel van PMML uitvoerbaar in de praktijk?

Voor deze concrete gevalstudie besluiten we dat het bekomen PMML model niet kan worden geïmplementeerd. Dit is te wijten aan het ontbreken van de componenten in het gegenereerde PMML document voor de beschrijving van getransformeerde variabelen. Hierdoor geeft het document geen bijkomende informatie over de transformaties die nodig zijn om de inputvariabelen van het model te berekenen. De doelomgeving kan dus niet de juiste gegevens voorzien voor de toepassing van het model. Indien de componenten voor transformaties echter wel aanwezig zouden zijn, kunnen we geen bijkomende redenen aanhalen die een mogelijke integratie in de weg zouden staan.

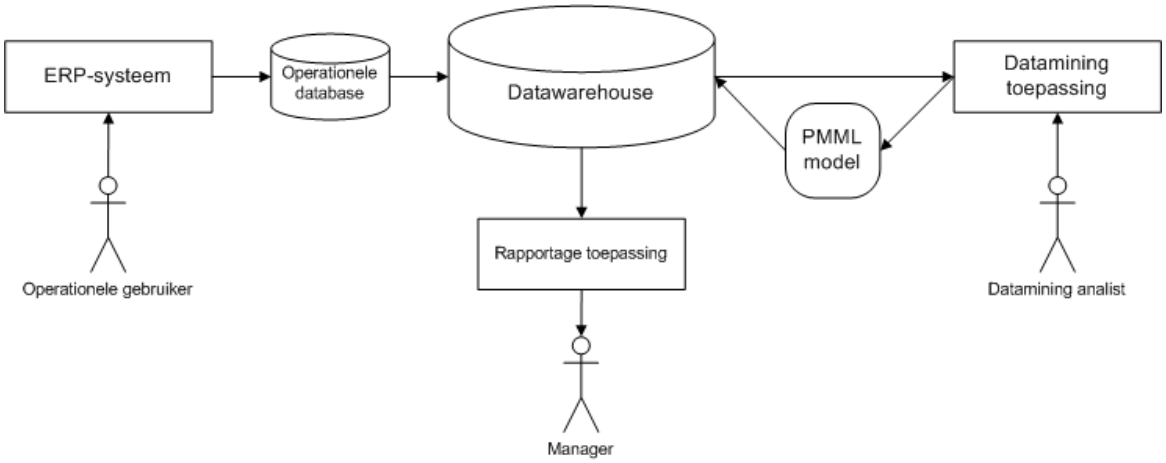
9.2 Algemene conclusie

Deze gevalstudie maakt uiteraard gebruik van een specifieke ERP-toepassing met zijn eigen mogelijkheden en beperkingen. Toch mogen we stellen dat deze representatief is voor andere ERP-systemen. Over het algemeen kunnen we stellen dat net die eigenschappen die eigen zijn aan ERP, essientiële hindernissen vormen voor analysetoepassingen zoals datamining. Hierbij denken we bijvoorbeeld aan het veranderlijke karakter van de operationele gegevens in ERP-systemen. Datamining daarentegen vereist een stabiele omgeving voor de ontwikkeling van een zo representatief mogelijk model. Als gevolg van de tekortkomingen van ERP-systemen zullen doorgaans aanvullende faciliteiten moeten worden ingericht om de gegevens beschikbaar te stellen voor datamining. Datawarehousing is hiervoor uitermate geschikt.

De structuur van een datawarehouse leent zich uitstekend voor het uitvoeren van multidimensionale analyses. De belangrijkste eigenschap is echter de mogelijkheid om heterogene gegevensbronnen te integreren en de toegang tot deze gegevens vlot en uniform te laten verlopen. Op die manier kunnen externe gegevens consistent worden geïntegreerd. Daarnaast creëert een datawarehouse een zeer stabiele omgeving, doordat men gegevens enkel kan raadplegen en niet wijzigen. Een datawarehouse bevat tenslotte niet alleen actuele gegevens, maar ook historische data die het mogelijk maken om trends en ontwikkelingen te identificeren. Door de integratie van een datawarehouse database met een datamining toepassing kan een goed model worden bekomen gebaseerd op de gegevens uit het ERP-systeem.

De voornaamste datamining toepassingen voorzien reeds de mogelijkheid om PMML modellen te genereren. Toch mogen we hieruit niet veronderstellen dat PMML modellen zonder meer kunnen worden geïntegreerd in andere toepassingen. Bepalend voor het al dan niet slagen van deze integratie is de PMML versie waarin het model is uitgedrukt en de mate waarin de hoofdcomponenten van het PMML document worden ondersteund. Doordat PMML een zeer recent ontwikkelde standaard is, bestaan er grote verschillen tussen opeenvolgende versies. Hierbij denken we bijvoorbeeld aan de overschakeling van DTD's in versie 2.0 naar XML schema's in versie 2.1. Dit maakt dat niet alle versies compatibel zijn en dat bij de keuze van een datamining oplossing hiermee rekening moet worden gehouden. Een ander belangrijk knelpunt is het niet ondersteunen van alle componenten van een PMML document door een datamining toepassing. Ook hier dient rekening mee te worden gehouden bij de ontwikkeling van een model.

Aan de hand van figuur 9.1, wordt de integratie van een datamining model in een ERP-systeem met behulp van PMML geïllustreerd.



Figuur 9.1: Integratie van een datamining model in een ERP-systeem

Een operationele gebruiker voert transactiegegevens in, in het ERP-systeem. Deze gegevens worden opgeslagen in de operationele database. Het datawarehouse extraheert deze gegevens op regelmatige tijdstippen, transformeert ze en slaat ze op in een nieuwe structuur. Een dataming analist maakt gebruik van een datamining toepassing om aan de hand van de gegevens uit het datawarehouse een model te ontwikkelen. Dit model wordt vervolgens omgezet naar PMML code en geëxporteerd naar het datawarehouse. Het datawarehouse vertaalt het geïmporteerde model naar een concrete functie, en voert deze functie uit voor de gegevens afkomstig uit het ERP-systeem. De resultaten worden tenslotte gerapporteerd aan het management.

De introductie van PMML opent heel wat nieuwe mogelijkheden. Door het scheiden van de creatie en de implementatie van datamining modellen, wordt het gebruik ervan aanzienlijk vereenvoudigd. Een datamining specialist kan instaan voor de ontwikkeling van de modellen, terwijl de klant de modellen op het gewenste tijdstip en op de gewenste gegevens kan toepassen in zijn eigen applicatie. Doordat PMML gebaseerd is op XML, zijn de geëxtraheerde modellen bovendien onafhankelijk van de toepassing, de implementatie, het hardwareplatform en het besturingssysteem.

Deel III

Bijlagen

Bijlage A

De tabel IART

IART Artikelbestand				
Naam van het veld	**Veldnr.**	**Omschrijving gegevensinhoud**	**Type**	**Lengte**
NUM	10	Artikelnr. (automatische nr.)	AutoIncrement	0
REF	20	Artikelreferentie	String	25
ISTITLE	30	Artikelgroep	Boolean	0
HEADING1	40	Artikelomschrijving	String	50
HEADING2	50	Artikelomschrijving (2 de taal)	String	50
HEADING3	55	Artikelomschrijving (3 de taal)	String	50
CAT	60	Artikelcategorie	String	4
DISCOUNT	70	Kortingspercentage	Float	0
DISCOUNTCAT	80	Kortingscategorie	String	6
UNIT	90	Eenheid	String	10
RANGETYPE	100	Hoofdartikel - type artikel (0: normal, 1,2,3: niveau)	String	1
ISMAIN	110	Wijziging van ingaven toegelaten	Boolean	0
MAINREF	120	Hoofdartikel - num ref.	String	25
MAINNUM	130	Hoofdartikel - num hoofd	Integer	0
GRID1	140	Hoofdartikel - type 1	String	10
GRID2	150	Hoofdartikel - type 2	String	10
GRID3	160	Hoofdartikel - type 3	String	10
GRIDORDER	170	Volgorde gamma's	String	3
GRIDSEP	175	Scheidingsteken	String	1
MODHEADING	180	Omschrijving wijzigbaar (Ja/nee)	Boolean	0
MODIMPUT	190	Wijziging van ingaven toegelaten	Boolean	0
NIMPUT	200	Nationale ingaven	String	10
EIMPUT	210	EEG ingaven	String	10
IIMPUT	220	Internationale ingaven	String	10
STOIMPUT	230	Ingaven voorraad	Float	0
VATNNAT1	240	Nationale BTW aard	String	3
VATNNAT2	250	Nationale BTW aard (2 de taal)	String	3
VATNCMP	260	Nationale BTW voet	Float	0
VATENAT1	270	EEG BTW aard	String	3
VATENAT2	280	EEG BTW aard (2de taal)	String	3
VATECMP	290	EEG BTW voet	Float	0
VATINAT1	300	Internationale BTW aard	String	3
VATINAT2	310	Internationale BTW aard (2de taal)	String	3
VATICMP	320	Internationale BTW voet	Float	0
SUPPID	330	Leveranciersreferentie	String	10
SUPPREF	340	Artikelreferentie bij de leverancier	String	30
PURPRICE	350	Aankoopprijs	Currency	0
PURCURRENCY	360	Aankoopvaluta	String	3
PURMINQUANT	370	Minimum aankoophoeveelheid	Float	0
PUROPTQUANT	380	Optimale aankoophoeveelheid	Float	0
PURDELIVDELAY	390	Leveringstermijn bij aankoop	Integer	0
PURLASTSUPP	400	Artikel voor intern gebruik	String	10
PURLASTPRICE	410	Niet wijzigbaar artikel (Ja/Nee)	Currency	0

PURLASTCURR	420	Memo	String	3
ISNATURE	430	Nationaal	String	2
ISINOUTCTRY	440	Niet nationaal	String	3
ISGOODSCODE	450	Statistieke goederencode	String	9
ISTRSPMEANS	460	Transport	String	1
ISBLOCK	470	Geblokkeerd artikel (Ja/Nee)	Boolean	0
ISWARNING	480	Gegarandeerd artikel (Ja/Nee)	Boolean	0
ISINTERNALUSE	490	Artikel voor intern gebruik	Boolean	0
ISREADONLY	500	Niet wijzigbaar artikel (Ja/Nee)	Boolean	0
MEMO	510	Memo	Memo	10
FAMILY	550	Artikelgroep	String	6
PURUNIT	560	Aankoopeenheid	String	4
SALUNIT	570	Verkoopeenheid	String	4
STOUNIT	580	Eenheid voorraad	String	4
STATUNIT	590	Supplementaire statistische eenheid	String	4
SERAFTERSAL	600	Hoeveelheid dienst na verkoop	Float	0
WEIGHTEDPR	630	Verkoopprijs samengesteld aritkel	Float	0
COSTPRCMP	640	Berekeningsformule prijs geheel	String	4
COSTPR	650	Aankoopprijs	Float	0
QTYCUSORD	670	Statistisch gewicht	Float	0
QTYSUPORD	680	Massa	Float	0
FIFO	690	Datum laatste bijwerking van de GG	Float	0
LEVVISI	700	Zichtbaarheidsniveau	Integer	0
LEVVISICMP	710	Zichtbaarheidsniveau samengesteld artikel	Integer	0
QTYINSTO	720	Nationale BTW natuur (Aankoop)	Float	0
ISNUMSERIAL	730	Serienummer (Ja/nee)	Boolean	0
ISSTO	740	In voorraad (Ja/Nee)	Boolean	0
SALPRMIN	750	Minimum verkoopprijs	Float	0
QTYONLOAN	760	EEG BTW natuur (2 de taal) (Aankoop)	Float	0
PRDEFAULT	770	Defaultprijs	Float	0
SALPRCMP	780	Verkoopprijs samengesteld aritkel	Float	0
PURIMPUT	790	Ingavenrekening aankoop	String	10
ARTREPLACE	800	Vervangartikel	String	25
QTYRESERVE	810	Nationale ingaven (Aankoop)	Float	0
STATUS	820	Normaal, vervallen, intern	Integer	0
STATWEIGHT	830	Statistisch gewicht	Float	0
VALUEPR	840	Naam variabele Bob.ini met naam directory	Float	0
VALUECMP	850	Bestandsnaam afbeelding	Float	0
DEFQTY	880	Defaulthoeveelheid bij de ingaven	Float	0
ISCLOSED	890	Gesloten	Boolean	0

WEIGHT	900	Massa	Float	0
ISSUPUNIT	910	Extra éénheden	Float	0
COSTPRTYP	920	Typeprijs retours	String	3
DLASTWPR	930	Datum laatste bijwerking van de GG	TimeStamp	0
QTYWPR	940	Hoeveelheid van de laatste bijwerking van de GG	Float	0
SHELVING	950	Rij	String	15
SVATNNAT1	960	Nationale BTW aard (Aankoop)	String	3
SVATNNAT2	970	Nationale BTW aard (2 de taal) (Aankoop)	String	3
SVATENAT1	980	EEG BTW aard (Aankoop)	String	3
SVATENAT2	990	EEG BTW aard (2 de taal) (Aankoop)	String	3
SVATINAT1	1000	Internationale BTW aard (Aankoop)	String	3
SVATINAT2	1010	Internationale BTW aard (2 de taal) (Aankoop)	String	3
SVATNCMP	1020	Nationale BTW voet (Aankoop)	Float	0
SVATECMP	1030	EEG BTW voet (Aankoop)	Float	0
SVATICMP	1040	Internationale BTW voet (Aankoop)	Float	0
SNIMPUT	1050	Nationale ingaven (Aankoop)	String	10
SEIMPUT	1060	EEG ingaven (Aankoop)	String	10
SIIMPUT	1070	Internationale ingaven (Aankoop)	String	10
EAN	1080	Code EAN	String	40
EANTYPE	1085	Type code EAN	String	10
PICTUREPATH	1090	Naam variabele Bob.ini met naam directory	String	20
PICTURENAME	1100	Bestandsnaam afbeelding	String	20
EXTRADISC	1110	Artikel met of zonder disconto en korting	Boolean	0
ASSPRICE	1120	Prijs geheel (ASSEM) of componenten (COMPO)	String	7
ASSSTKUPDT	1130	Update voorraad geheel (ASSEM) of componenten (COMPO)	String	7
ASSOPTIONS	1140	Opties zichtbaarheid of componenten	String	4
ASSEMBLED	1150	Samengesteld artikel of niet	Boolean	0
FIXPRICE	1160	Vaste prijs of niet	Boolean	0
FIXPRICEAMN	1170	Bedrag vaste prijs	Currency	0
FIXHEADDISC	1180	In rekening nemen van de korting hoofding	Boolean	0
FIXLINEDISC	1190	In rekening nemen van de korting detail	Boolean	0
CREATEDBY	1210	Auteur	String	10
CREATEDON	1220	Aanmaakdatum	TimeStamp	0
MODIFIEDBY	1230	Auteur laatste wijziging	String	10
MODIFIEDON	1240	Datum laatste wijziging	TimeStamp	0
TRFTSTATUS	1250	Status transfer	String	3
LABNBLAB	1270	Aantal etiketten per reeks	Integer	0
LABPACK	1280	Reeks	Float	0
LABLAYOUTIMP	1290	Lay-out artikel import	String	5
LABLAYOUTEXP	1300	Lay-out etiket export	String	5
LABLAYOUTART	1305	Lay-out etiket artikel	String	5
ISDIRECTORDER	1310	Artikel in directe bestelling	Boolean	0
LINKED	1320	Verbonden artikel	Boolean	0
LINKTOSAL	1330	Verbonden artikel verkopen	Boolean	0
LINKTOPUR	1340	Verbonden artikel aankopen	Boolean	0

Figuur A.1: De tabel IART

Bijlage B

De tabel COMPAN

COMPAN Gegevens van klanten/leveranciers/prospecten				
Naam van het veld	**Veldnr.**	**Omschrijving gegevensinhoud**	**Type**	**Lengte**
CID	10	Bedrijfsreferentie (derden)	String	10
CCUSTYPE	20	Type klant (C: klant, U geen)	String	1
CSUPTYPE	30	Type leverancier (S: leverancier, U geen)	String	1
CNAME1	40	Naam 1 derden	String	40
CNAME2	50	Naam 2 derden	String	40
CADDRESS1	60	Adres 1 derden	String	40
CADDRESS2	70	Adres 2 derden	String	40
CZIPCODE	80	Postcode	String	10
CSVATNO	90	Niet gebruikt	String	15
CLOCALITY	100	Plaats	String	40
CLANGUAGE	110	Taalcode	String	2
CISPERS	120	Fysische persoon	Boolean	0
CPROCAT	130	Prospectcategorie	String	3
CSUPCAT	140	Leverancierscategorie	String	3
CCUSCAT	150	Klantcategorie	String	3
CCURRENCY	160	Valutacode	String	3
CVATREF	170	BTW referentie	String	2
CVATNO	180	BTW nummer	String	12
CVATCAT	190	BTW categorie (' ': onderworpen, N: Niet onderworpen, X: Uitzondering)	String	1
CTELNO	200	Telefoonnummer	String	14
CFAXNO	210	Faxnummer	String	14
CMEMO	220	Memo	Memo	1
CCUSLSTMNO	230	Volgende afpuntnummer te gebruiken voor klanten	Integer	0
CSUPLSTMNO	240	Volgende afpuntnummer te gebruiken voor leveranciers	Integer	0
CCUSVNAT1	250	Aard BTW klant per default (Ingaven)	String	3
CCUSVNAT2	260	Aard BTW klant per default (Ingaven)	String	3
CCUSVATCMP	270	BTW voet klant per default (Ingaven)	Float	0
CSUPVNAT1	280	Aard BTW leverancier per default (Ingaven)	String	3
CSUPVNAT2	290	Aard BTW leverancier per default (Ingaven)	String	3
CSUPVATCMP	300	BTW voet leverancier per default (Ingaven)	Float	0
CCUSCTRACC	310	Collectieve klantenrekeningen	String	10
CSUPCTRACC	320	Collectieve leveranciersrekeningen	String	10
CCUSIMPUTA	330	Ingaverekening klant per default (Ingaven)	String	10
CSUPIMPUTA	340	Ingaverekening leverancier per default (Ingaven)	String	10
CSUPCATCOMM	350	Aard Commissiefiches, fiche (281.50)	String	1
CCTRYCODE	360	ISO code land van de onderneming	String	2
CBANKCODE	370	Code inter-bank	String	8
CBANKNO	380	Bankrekeningnummer	String	20
DBNKQUALIF	390	Te wissen	String	8
CISWARNING	400	True = Waarschuwing t.g.v. de ingaven	Boolean	0
CISREADONL	410	True = Ingave verboden (klant/leverancier)	Boolean	0
CISBLOCK	420	True = Geblokkeerde klant	Boolean	0

CISSECRET	430	True = Historieken onzichtbaar behalve check	Boolean	0
CCUSPAYDELAY	440	Code betalingstermijn per default (Ingaven)	String	6
CSUPPAYDISC	450	Kortingspercentage	Float	0
CSUPPDISCDEL	460	Kortingstermijn (in dagen)	Integer	0
CTEMPCUSDEBIT	470	Debetbedrag klant (tijdelijk)	Currency	0
CTEMPCUSCREDIT	480	Creditbedrag klant (tijdelijk)	Currency	0
CTEMPSUPDEBIT	490	Debetbedrag leverancier (tijdelijk)	Currency	0
CTEMPSUPCREDIT	500	Creditbedrag leverancier (tijdelijk)	Currency	0
CREMLASTDATE	510	Datum laatste aanmaning	TimeStamp	0
CREMLASTLEVEL	520	Niveau laatste aanmaning	Integer	0
CREMCAT	530	Categorie aanmaning	String	5
CREMSTATUS	540	Satus aanmaning	String	1
CTOTINTEREST	550	Totaal verschuldigde intresten	Currency	0
AUTHOR	580	Persoon die de boodschap aangemaakt of laatst gewijzigd heeft	String	10
CNATREGISTRYID	620	Inschrijvingsnummer nationaal register "99.99.99 999-99"	String	20
CSUPPAYDELAY	630	Code betalingstermijn par default (Ingaven)	String	6
CMANUALPAY	640	Handleiding betaling (geen bankverbinding)	Boolean	0
CINDEPPAYEE	650	Diverse begunstigden	Boolean	0
CINDEPPIMPUT	660	Ingaverekeningen diverse begunstigden	String	10
CBNKCIVILITY	670	Aansprekingscode voor de bank correspondentie	String	1
CBNKLANGCODE	680	Taalcode voor de bank correspondentie	String	1
CBNKBANKDBK	690	Betalingsdagboek leverancier	String	4
CFACTORING	700	Betaling factoring maatschappij	Boolean	0
CFACTORID	710	Referentie factorfiche	String	10
CBANKORDERPAY	720	Betaling domiciliëring	Boolean	0
CBANKORDERPAYNO	730	Domiliciëringsnummer	String	15
CCUSPDISCDEL	740	Kortingstermijn (in dagen)	Integer	0
CCUSPAYDISC	750	Kortingspercentage	Float	0
CCUSTEMPLID	760	Referentie sjabloon klant	String	10
CSUPTEMPLID	770	Referentie sjabloon leverancier	String	10
EMAILADDRESS	780	E-mail adres	String	60
HTTPADDRESS	790	Internet adres	String	60
CSUPPJOB	800	Beroep uitgeoefend door de leverancier voor de commissiefiches	String	10
CREATEDBY	810	Auteur	String	10
CREATEDON	820	Aanmaakdatum	TimeStamp	0
MODIFIEDBY	830	Auteur laatste wijziging	String	10
MODIFIEDON	840	Datum laatste wijziging	TimeStamp	0
TRFTSTATUS	850	Status transfer	String	3
STATIONID	860	Referentie werkpost	String	3
CBUSINESSNO	870	Ondernemingsnummer	String	20
CSLEEPING	880	K/L in slaapstand	Boolean	0

Figuur B.1: De tabel COMPAN

Bijlage C

De tabel IHISTO

IHISTO Historiek van artikelbewegingen				
Naam van het veld	**Veldnr.**	**Omschrijving gegevensinhoud**	**Type**	**Lengte**
DBK	10	Dagboekcode	String	4
FYEAR	20	Fiscaal jaar	String	5
DOCNO	30	Documentnummer	Integer	0
DOCLINE	40	Documentlijnnummer	Integer	0
YEAR	50	Jaar	Integer	0
MONTH	60	Maand	Integer	0
DOCDATE	70	Datum document	Date	0
CPID	80	Referentie derde	String	10
DBKTYPE	90	Dagboektype	String	4
LINETYPE	100	Lijntype	String	1
ARTNUM	110	Artikelnummer	Integer	0
ARTREF	120	Artikelreferentie	String	25
IMPUT	130	Ingavenrekening	String	10
DOCSTATUS	140	Status document	String	1
DOCLINKNO	150	Verbonden documentnummer	Integer	0
DOCLINKLINE	160	Verbonden lijndocumentnummer	Integer	0
QTYMVT	170	Hoeveelheid en eenheid van voorraad	Float	0
QTYORDER	180	Bestelde hoeveelheid	Float	0
QTYDELIV	190	Geleverde hoeveelheid	Float	0
QTYINVABLE	200	Beschikbare hoeveelheid	Float	0
QTYINVOICE	210	Factuurhoeveelheid	Float	0
QTYINT1	220	Interne hoeveelheid 1	Float	0
COMMENT	230	Commentaarlijn	String	120
TYPEVALUE	240	Geldigheidstype	String	1
VSTORED	250	BTW code (unieke sleutel)	String	10
TVATTYPE	260	Type BTW code	String	1
TVATNAT1	270	Aard BTW code	String	3
TVATNAT2	280	Aard 2 BTW code	String	3
TVATCMP	290	BTW voet	Float	0
TVATOUTDAT	300	BTW buiten periode	Boolean	0
WAREHOUSE	310	Magazijn	String	21
CURRENCY	320	Valuta	String	3
WEIGHTEDPR	330	Gemiddelde gewogen prijs	Float	0
PACKING	340	Voorwaardelijke coëfficient	Float	0
PUPRLIST	350	Eenheidsprijs van prijslijst	Currency	0
PU	360	Eenheidsprijs	Currency	0

PRCDISC	370	Disconto %	Float	0
BASEAMN	380	Basis BTW in basisvaluta	Currency	0
CURRBASEAMN	390	Basis BTW in facturatievaluta	Currency	0
PAYAMN	400	Te betalen bedrag (BTW incl.) in basisvaluta	Currency	0
CURRPAYAMN	410	Te betalen bedrag (BTW incl.) in factuurvaluta	Currency	0
LINKDBK	420	Verbonden dagboekcode van brondocument	String	4
LINKFYEAR	430	Fiscaal boekjaar gelinkt document	String	5
LINKDOCNO	440	Verbonden documentnummer van doeldocument	Integer	0
LINKNO	450	Linknummer	Integer	0
ORIDBK	460	Dagkboekcode van origineel getransfereerd document	String	4
ORIFYEAR	470	Fiscaal jaar van origineel getransfereerd document	String	5
ORIDOCNO	480	Nummer van origineel getransfereerd document	Integer	0
ORIDOCLINE	490	Lijnnummer van origineel getransfereerd document	Integer	0
ISBOLD	500	Commentaarlijn in BOLD	Boolean	0
ISUNDERLINE	510	Commentaarlijn in Underline	Boolean	0
TVATINST	520	Voet die tussenkomt in de intrastat	Boolean	0
CONVCLASS	530	Niet in gebruik	String	1
INTUSE1	540	Gebruik intern 1	Integer	0
LINESTATUS	550	Status detaillijn	String	1
CPTYPE	560	Derdentype	String	1
DELIVDATE	570	Leveringsdatum	TimeStamp	0
WEIGHT	580	Gewicht	Float	0
SHELVING	590	Rij	String	15
EXTRADISC	600	Korting of niet	Boolean	0
ARTASSLINE	610	Samenstellingnummer als samenstelling	Integer	0
ARTCOMPLINE	620	Samenstellingsnummer als artikel	Integer	0
ASSVISIBLE	630	Zichtbaarheid lijn	Boolean	0
ASSSTKUPDT	640	Voorraadbeheer	Boolean	0
ASSVALOR	650	Waardering lijn	Boolean	0
PRCODE	660	Code van specifieke lijst	String	10
PRVERSION	670	Geldige periode van specifieke lijst	Integer	0
PRTYPE	680	Specifieke prijslijst voor bronklant	String	8
ISTBLDISC	690	Kortingstabel in rekening brengen	Boolean	0
ISDISC	700	Kortingen in rekening brengen	Boolean	0
VATIAMN	710	Bedrag in BTW in basisvaluta	Currency	0
CURRVATIAMN	720	Bedrag in BTW in basisvaluta	Currency	0
NETPU	740	Netto eenheidsprijs	Currency	0
ISDIRECTORDER	750	Artikel onder directe bestelling	Boolean	0
DISCAMN	760	Korting in Euro	Currency	0
CURRDISCAMN	770	Korting in Euro	Currency	0
CTCID	780	Contractcode	String	10

Figuur C.1: De tabel IHISTO

Bijlage D

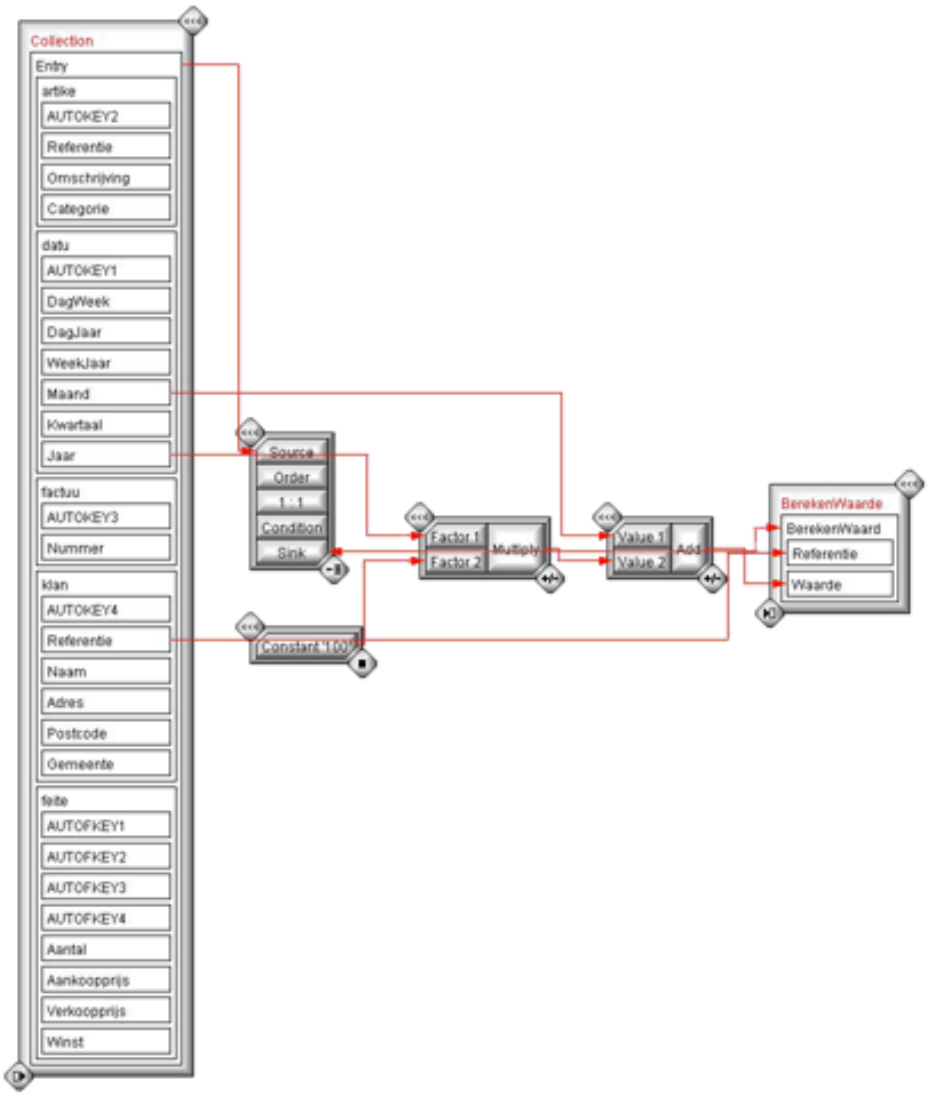
SQL-declaraties van de datamart

```
CREATE DATABASE IF NOT EXISTS Datamart;
USE Datamart;
DROP TABLE IF EXISTS Datum;
CREATE TABLE Datum
(
        AUTOKEY1        BIGINT  NOT NULL,
        DagWeek         VARCHAR(20) NOT NULL,
        DagJaar         INTEGER NOT NULL,
        WeekJaar        INTEGER NOT NULL,
        Maand           INTEGER NOT NULL,
        Kwartaal        INTEGER NOT NULL,
        Jaar            INTEGER NOT NULL,
        PRIMARY KEY (
                AUTOKEY1, DagJaar, Jaar
        )
);
DROP TABLE IF EXISTS Artikel;
CREATE TABLE Artikel
(
        AUTOKEY2        BIGINT  NOT NULL,
        Referentie      VARCHAR(25) NOT NULL,
        Omschrijving    VARCHAR(50) NOT NULL,
        Categorie       VARCHAR(4) NOT NULL,
        PRIMARY KEY (
                AUTOKEY2, Referentie
        )
);
```

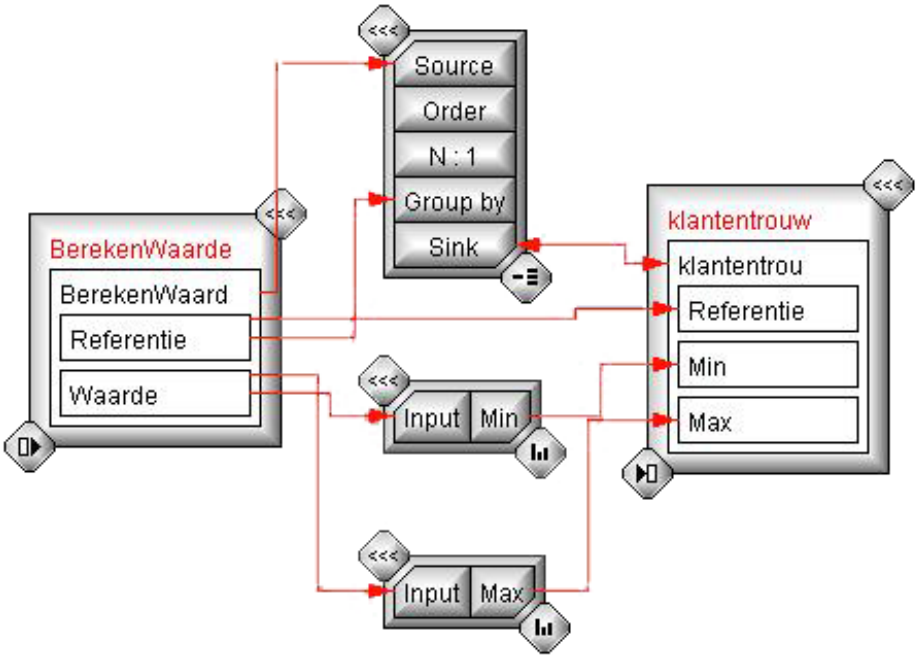
```
DROP TABLE IF EXISTS Factuur;
CREATE TABLE Factuur
(
        AUTOKEY3        BIGINT  NOT NULL,
        Nummer          INTEGER NOT NULL,
        PRIMARY KEY (
                AUTOKEY3, Nummer
        )
);
DROP TABLE IF EXISTS Klant;
CREATE TABLE Klant
(
        AUTOKEY4        BIGINT  NOT NULL,
        Referentie      VARCHAR(25) NOT NULL,
        Naam            VARCHAR(50) NOT NULL,
        Adres           VARCHAR(50) NOT NULL,
        Postcode        VARCHAR(25) NOT NULL,
        Gemeente        VARCHAR(50) NOT NULL,
        PRIMARY KEY (
                AUTOKEY4, Referentie
        )
);
DROP TABLE IF EXISTS Feiten;
CREATE TABLE Feiten
(
        AUTOFKEY1       BIGINT  NOT NULL,
        AUTOFKEY2       BIGINT  NOT NULL,
        AUTOFKEY3       BIGINT  NOT NULL,
        AUTOFKEY4       BIGINT  NOT NULL,
        Aantal          INTEGER DEFAULT 0,
        Aankoopprijs    DECIMAL(14,4)   DEFAULT 0.0,
        Verkoopprijs    DECIMAL(14,4)   DEFAULT 0.0,
        Winst           DECIMAL(14,4)   DEFAULT 0.0
);
```

Bijlage E

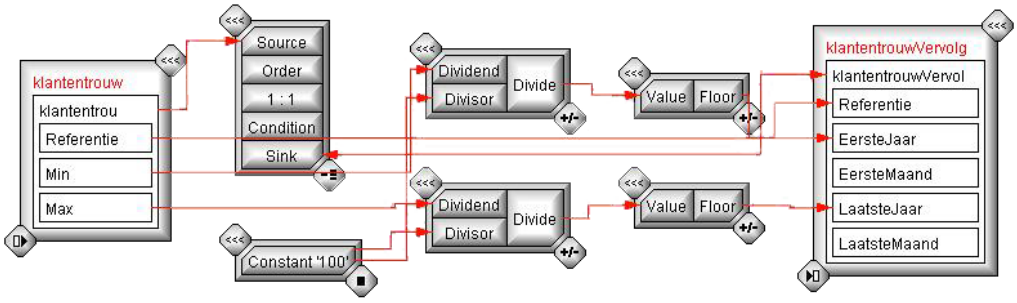
Visuele voorstellingen van de transformaties in ETL4ALL voor het berekenen van de variabelen



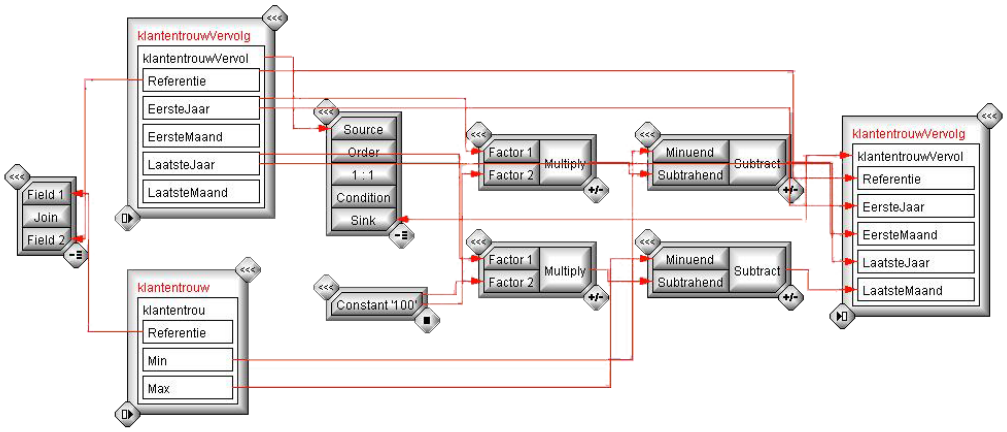
Figuur E.1: De procedure *berekenWaarde*



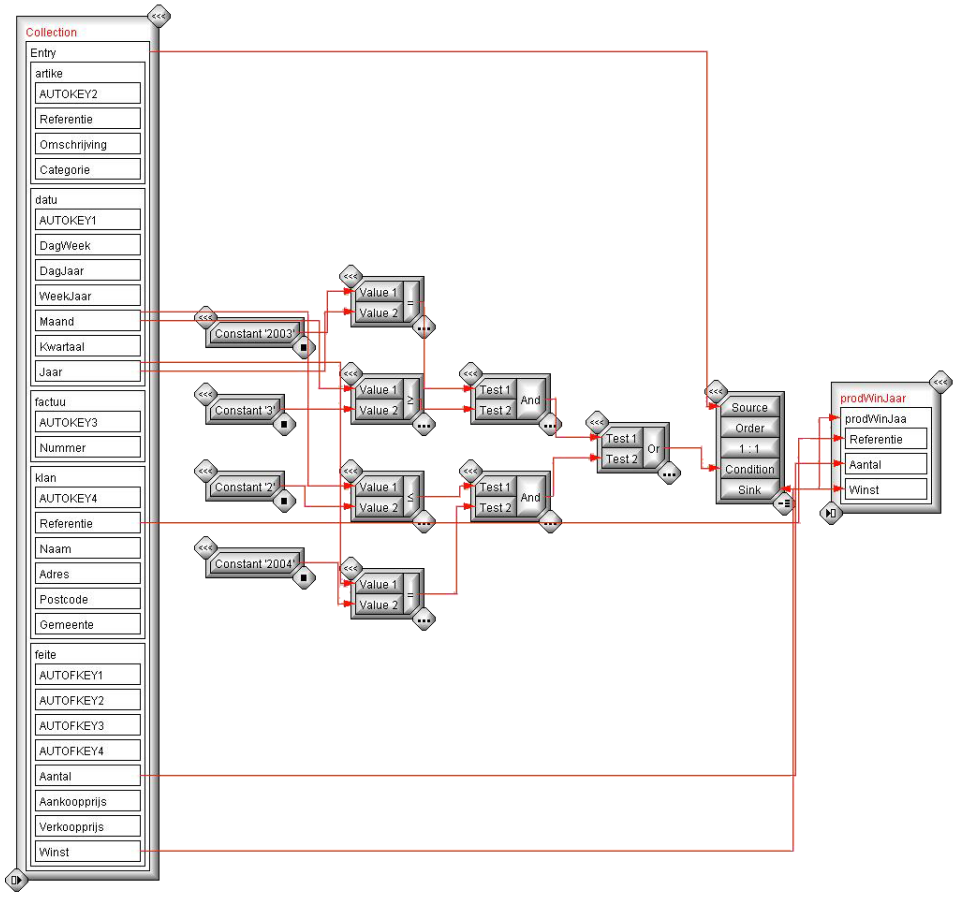
Figuur E.2: De procedure *berekenTrouwKlant*



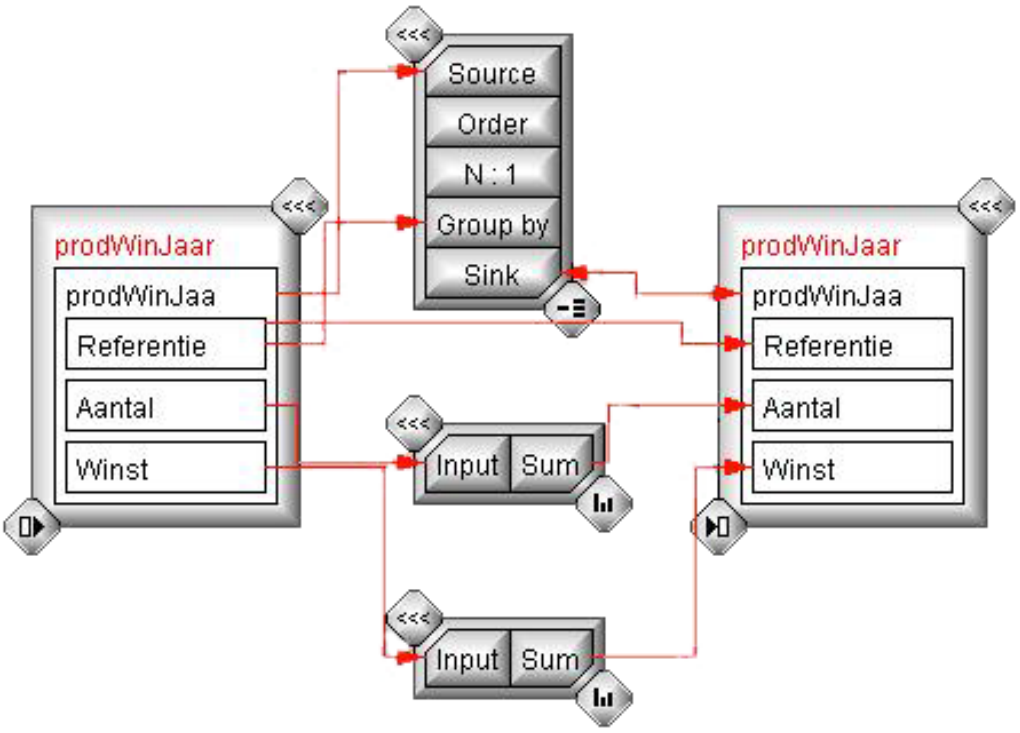
Figuur E.3: De procedure *berekenTrouwKlantVervolg*



Figuur E.4: De procedure *berekenTrouwKlantVervolg2*



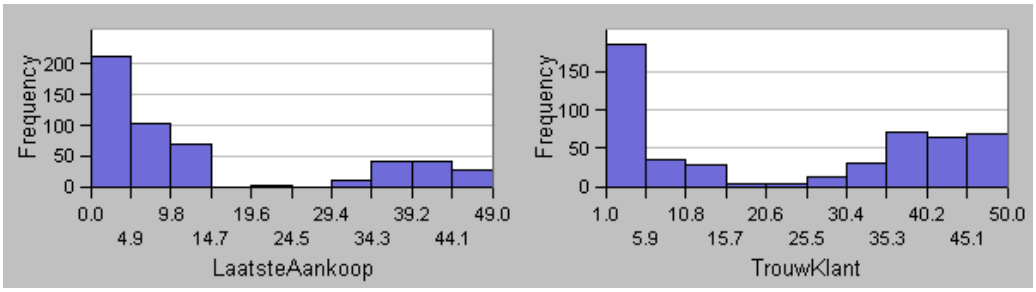
Figuur E.5: De procedure *berekenProdWinJaar*



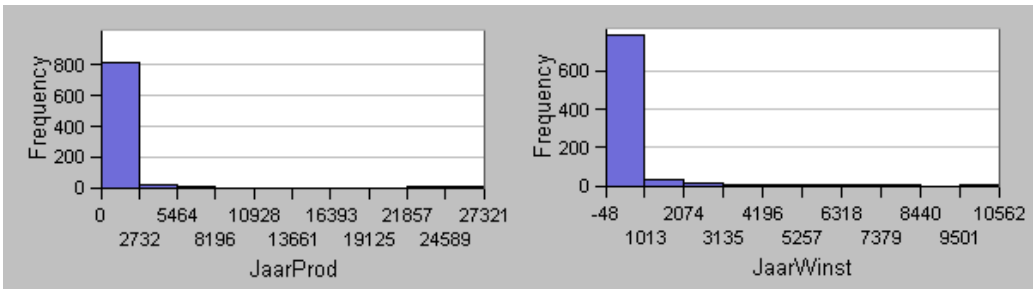
Figuur E.6: De procedure *berekenSom*

Bijlage F

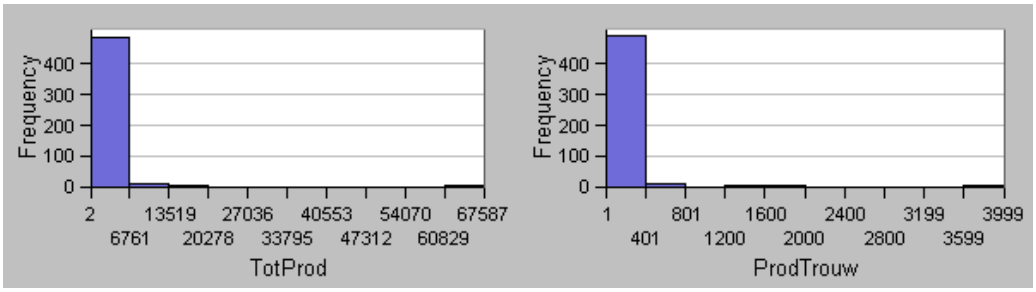
Grafische analyse van de variabelen



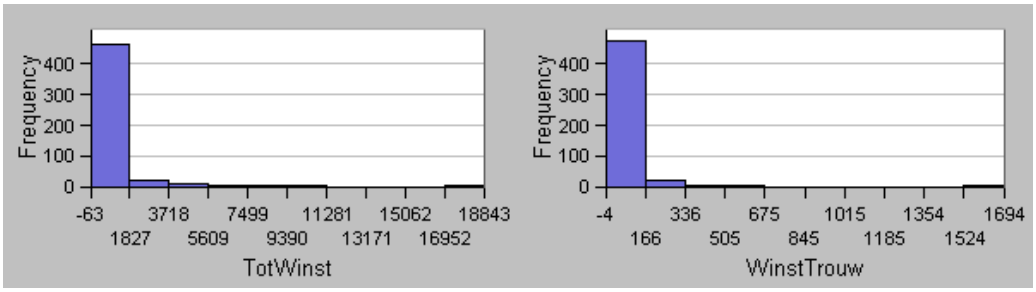
Figuur F.1: De variabelen LaatsteAankoop en TrouwKlant



Figuur F.2: De variabelen JaarProd en JaarWinst



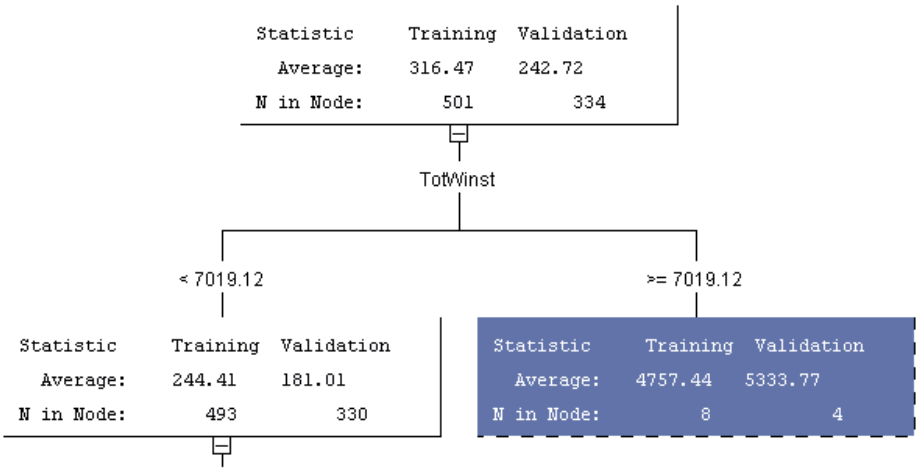
Figuur F.3: De variabelen TotProd en ProdTrouw



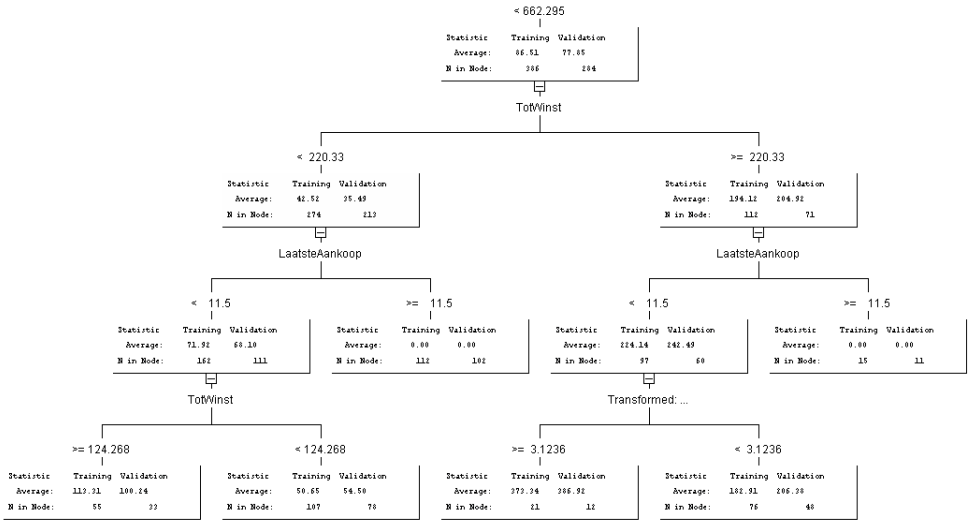
Figuur F.4: De variabelen TotWinst en WinstTrouw

Bijlage G

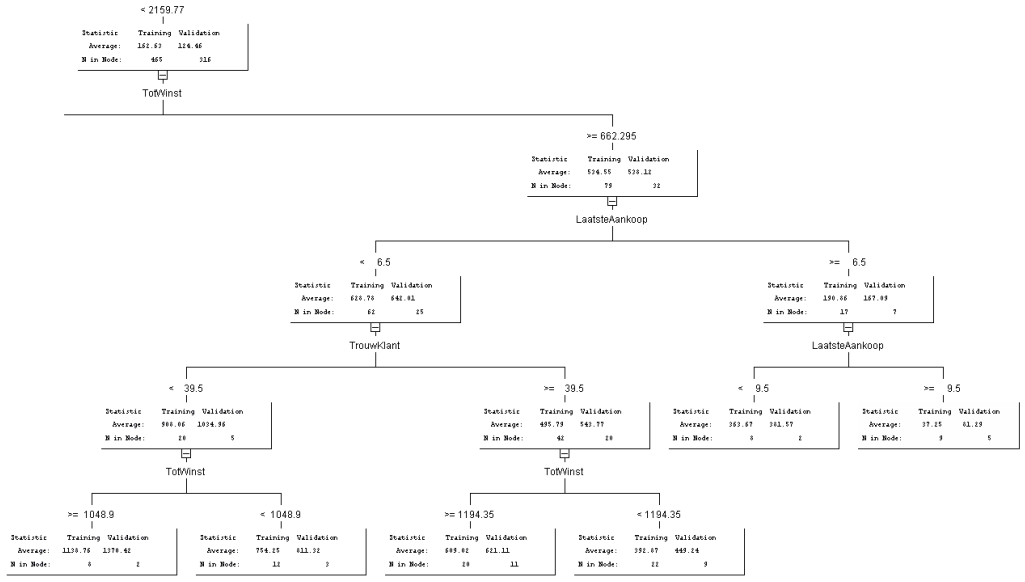
Uitgebreide voorstelling van beslissingsboom vijf



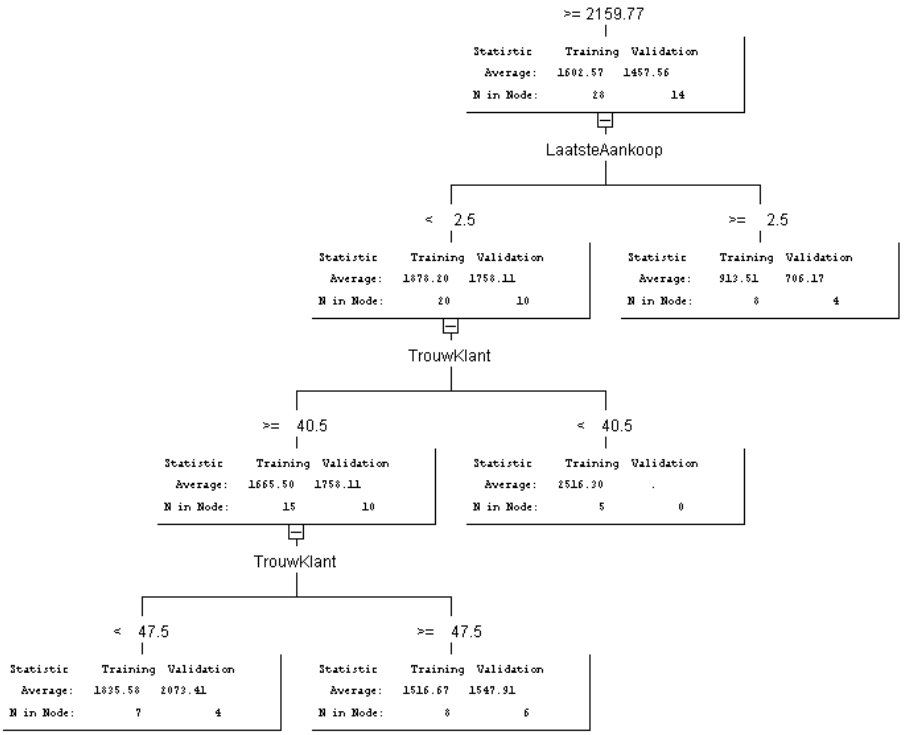
Figuur G.1: De top van de beslissingsboom



Figuur G.2: De linkse vertakkingen van de beslissingsboom



Figuur G.3: De middelste vertakkingen van de beslissingsboom



Figuur G.4: De rechtse vertakkingen van de beslissingsboom

Bijlage H

Het PMML document van beslissingsboom vijf

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
  <PMML version="2.1">
    <Header copyright="Copyright(c) 2002 SAS Institute Inc.,
    Cary, NC, USA. All Rights Reserved.">
    <Application name="SAS(r)" version="9.1"/>
    <Timestamp>2005-05-07 17:19:05</Timestamp>
    </Header>
    <DataDictionary numberOfFields="8">
      <DataField name="JaarWinst" x-SASlabel="JaarWinst"
      optype="continuous" dataType="double"/>
      <DataField name="LOG_ProdTrouw" x-SASlabel="Transformed: ProdTrouw"
      optype="continuous" dataType="double"/>
      <DataField name="LOG_TotProd" x-SASlabel="Transformed: TotProd"
      optype="continuous" dataType="double"/>
      <DataField name="LaatsteAankoop" x-SASlabel="LaatsteAankoop"
      optype="continuous" dataType="double"/>
      <DataField name="TotWinst" x-SASlabel="TotWinst"
      optype="continuous" dataType="double"/>
      <DataField name="TrouwKlant" x-SASlabel="TrouwKlant"
      optype="continuous" dataType="double"/>
      <DataField name="WinstTrouw" x-SASlabel="WinstTrouw"
      optype="continuous" dataType="double"/>
      <DataField name="Regio" x-SASlabel="Regio" optype="categorical"
      dataType="string" x-SASformat="$4." x-SASinformat="$."/>
```

```
</DataDictionary>
<TransformationDictionary/>
<TreeModel functionName="regression" splitCharacteristic="binarySplit">
  <MiningSchema>
    <MiningField name="JaarWinst" usageType="predicted"
    optype="continuous"/>
    <MiningField name="TotWinst" usageType="active"
    optype="continuous"/>
    <MiningField name="LaatsteAankoop" usageType="active"
    optype="continuous"/>
    <MiningField name="TrouwKlant" usageType="active"
    optype="continuous"/>
    <MiningField name="LOG_ProdTrouw" usageType="active"
    optype="continuous"/>
  </MiningSchema>
  <Extension>
    <Output>
      <OutputField name="P_JaarWinst" displayName="Predicted: JaarWinst"
      optype="continuous" dataType="double" targetField="JaarWinst"
      feature="predictedValue"/>
      <OutputField name="V_JaarWinst" displayName="Validated: JaarWinst"
      optype="continuous" dataType="double" targetField="JaarWinst"
      feature="predictedValue"/>
      <OutputField name="R_JaarWinst" displayName="Residual: JaarWinst"
      optype="continuous" dataType="double" targetField="JaarWinst"
      feature="residual"/>
    </Output>
  </Extension>
  <Targets>
    <Target field="JaarWinst" optype="continuous">
      <TargetValue defaultValue="316.474367465068"/>
    </Target>
  </Targets>
  <Node id="1" score="316.474367465" recordCount="501">
    <True/>
    <Node id="3" score="4757.4353" recordCount="8">
      <CompoundPredicate booleanOperator="surrogate">
        <SimplePredicate field="TotWinst" operator="greaterOrEqual"
```

```
    value="7019.12"/>
    <False/>
  </CompoundPredicate>
</Node>
<Node id="2" score="244.410092697" recordCount="493">
  <CompoundPredicate booleanOperator="surrogate">
    <SimplePredicate field="TotWinst" operator="lessThan"
    value="7019.12"/>
    <SimplePredicate field="TotWinst" operator="isMissing"/>
  </CompoundPredicate>
  <Node id="5" score="1602.57187857" recordCount="28">
    <CompoundPredicate booleanOperator="surrogate">
      <SimplePredicate field="TotWinst"
      operator="greaterOrEqual" value="2159.77"/>
      <False/>
    </CompoundPredicate>
    <Node id="9" score="913.508675" recordCount="8">
      <CompoundPredicate booleanOperator="surrogate">
        <SimplePredicate field="LaatsteAankoop"
        operator="greaterOrEqual"  value="    2.5"/>
        <False/>
      </CompoundPredicate>
    </Node>
    <Node id="8" score="1878.19716" recordCount="20">
      <CompoundPredicate booleanOperator="surrogate">
        <SimplePredicate field="LaatsteAankoop" operator="lessThan"
        value="    2.5"/>
        <SimplePredicate field="LaatsteAankoop" operator="isMissing"/>
      </CompoundPredicate>
      <Node id="14" score="2516.30144" recordCount="5">
        <CompoundPredicate booleanOperator="surrogate">
          <SimplePredicate field="TrouwKlant" operator="lessThan"
          value="   40.5"/>
          <False/>
        </CompoundPredicate>
      </Node>
      <Node id="15" score="1665.49573333" recordCount="15">
        <CompoundPredicate booleanOperator="surrogate">
```

```
          <SimplePredicate field="TrouwKlant"
          operator="greaterOrEqual" value="   40.5"/>
          <SimplePredicate field="TrouwKlant" operator="isMissing"/>
        </CompoundPredicate>
        <Node id="24" score="1835.58307142" recordCount="7">
          <CompoundPredicate booleanOperator="surrogate">
            <SimplePredicate field="TrouwKlant" operator="lessThan"
            value="   47.5"/>
            <False/>
          </CompoundPredicate>
        </Node>
        <Node id="25" score="1516.6693125" recordCount="8">
          <CompoundPredicate booleanOperator="surrogate">
            <SimplePredicate field="TrouwKlant"
            operator="greaterOrEqual" value="   47.5"/>
            <SimplePredicate field="TrouwKlant" operator="isMissing"/>
          </CompoundPredicate>
        </Node>
      </Node>
    </Node>
  </Node>
  <Node id="4" score="162.628307741" recordCount="465">
    <CompoundPredicate booleanOperator="surrogate">
      <SimplePredicate field="TotWinst" operator="lessThan"
      value="2159.77"/>
      <SimplePredicate field="TotWinst" operator="isMissing"/>
    </CompoundPredicate>
    <Node id="7" score="534.545692405" recordCount="79">
      <CompoundPredicate booleanOperator="surrogate">
        <SimplePredicate field="TotWinst"
        operator="greaterOrEqual" value="662.295"/>
        <False/>
      </CompoundPredicate>
      <Node id="13" score="190.858241176" recordCount="17">
        <CompoundPredicate booleanOperator="surrogate">
          <SimplePredicate field="LaatsteAankoop"
          operator="greaterOrEqual" value="    6.5"/>
          <False/>
```

```
    </CompoundPredicate>
    <Node id="22" score="363.6713375" recordCount="8">
      <CompoundPredicate booleanOperator="surrogate">
        <SimplePredicate field="LaatsteAankoop" operator="lessThan"
        value="    9.5"/>
        <False/>
      </CompoundPredicate>
    </Node>
    <Node id="23" score="37.2466" recordCount="9">
      <CompoundPredicate booleanOperator="surrogate">
        <SimplePredicate field="LaatsteAankoop"
        operator="greaterOrEqual" value="    9.5"/>
        <SimplePredicate field="LaatsteAankoop" operator="isMissing"/>
      </CompoundPredicate>
    </Node>
  </Node>
  <Node id="12" score="628.782574193" recordCount="62">
    <CompoundPredicate booleanOperator="surrogate">
      <SimplePredicate field="LaatsteAankoop" operator="lessThan"
      value="    6.5"/>
      <SimplePredicate field="LaatsteAankoop" operator="isMissing"/>
    </CompoundPredicate>
    <Node id="20" score="908.057535" recordCount="20">
      <CompoundPredicate booleanOperator="surrogate">
        <SimplePredicate field="TrouwKlant" operator="lessThan"
        value="   39.5"/>
        <False/>
      </CompoundPredicate>
      <Node id="31" score="1138.7642125" recordCount="8">
        <CompoundPredicate booleanOperator="surrogate">
          <SimplePredicate field="TotWinst"
          operator="greaterOrEqual" value=" 1048.9"/>
          <False/>
        </CompoundPredicate>
      </Node>
      <Node id="30" score="754.253083333" recordCount="12">
        <CompoundPredicate booleanOperator="surrogate">
          <SimplePredicate field="TotWinst" operator="lessThan"
```

```
            value=" 1048.9"/>
            <SimplePredicate field="TotWinst" operator="isMissing"/>
          </CompoundPredicate>
        </Node>
      </Node>
      <Node id="21" score="495.794497619" recordCount="42">
        <CompoundPredicate booleanOperator="surrogate">
          <SimplePredicate field="TrouwKlant"
          operator="greaterOrEqual" value="   39.5"/>
          <SimplePredicate field="TrouwKlant" operator="isMissing"/>
        </CompoundPredicate>
        <Node id="33" score="609.015145" recordCount="20">
          <CompoundPredicate booleanOperator="surrogate">
            <SimplePredicate field="TotWinst"
            operator="greaterOrEqual" value="1194.35"/>
            <False/>
          </CompoundPredicate>
        </Node>
        <Node id="32" score="392.866636363" recordCount="22">
          <CompoundPredicate booleanOperator="surrogate">
            <SimplePredicate field="TotWinst" operator="lessThan"
            value="1194.35"/>
            <SimplePredicate field="TotWinst" operator="isMissing"/>
          </CompoundPredicate>
        </Node>
      </Node>
    </Node>
  </Node>
  <Node id="6" score="86.5105010362" recordCount="386">
    <CompoundPredicate booleanOperator="surrogate">
      <SimplePredicate field="TotWinst" operator="lessThan"
      value="662.295"/>
      <SimplePredicate field="TotWinst" operator="isMissing"/>
    </CompoundPredicate>
    <Node id="11" score="194.118823214" recordCount="112">
      <CompoundPredicate booleanOperator="surrogate">
        <SimplePredicate field="TotWinst"
        operator="greaterOrEqual" value=" 220.33"/>
```

```
        <False/>
      </CompoundPredicate>
      <Node id="19" score="0" recordCount="15">
        <CompoundPredicate booleanOperator="surrogate">
          <SimplePredicate field="LaatsteAankoop"
          operator="greaterOrEqual" value="   11.5"/>
          <False/>
        </CompoundPredicate>
      </Node>
      <Node id="18" score="224.137197938" recordCount="97">
        <CompoundPredicate booleanOperator="surrogate">
          <SimplePredicate field="LaatsteAankoop" operator="lessThan"
          value="   11.5"/>
          <SimplePredicate field="LaatsteAankoop" operator="isMissing"/>
        </CompoundPredicate>
        <Node id="29" score="373.343271428" recordCount="21">
          <CompoundPredicate booleanOperator="surrogate">
            <SimplePredicate field="LOG_ProdTrouw"
            operator="greaterOrEqual" value=" 3.1236"/>
            <False/>
          </CompoundPredicate>
        </Node>
        <Node id="28" score="182.909203947" recordCount="76">
          <CompoundPredicate booleanOperator="surrogate">
            <SimplePredicate field="LOG_ProdTrouw" operator="lessThan"
            value=" 3.1236"/>
            <SimplePredicate field="LOG_ProdTrouw" operator="isMissing"/>
          </CompoundPredicate>
        </Node>
      </Node>
    </Node>
    <Node id="10" score="42.5246175182" recordCount="274">
      <CompoundPredicate booleanOperator="surrogate">
        <SimplePredicate field="TotWinst" operator="lessThan"
        value=" 220.33"/>
        <SimplePredicate field="TotWinst" operator="isMissing"/>
      </CompoundPredicate>
      <Node id="17" score="0" recordCount="112">
```

```
                <CompoundPredicate booleanOperator="surrogate">
                  <SimplePredicate field="LaatsteAankoop"
                  operator="greaterOrEqual" value="   11.5"/>
                  <False/>
                </CompoundPredicate>
              </Node>
              <Node id="16" score="71.9243530864" recordCount="162">
                <CompoundPredicate booleanOperator="surrogate">
                  <SimplePredicate field="LaatsteAankoop" operator="lessThan"
                  value="   11.5"/>
                  <SimplePredicate field="LaatsteAankoop" operator="isMissing"/>
                </CompoundPredicate>
                <Node id="27" score="113.307058181" recordCount="55">
                  <CompoundPredicate booleanOperator="surrogate">
                    <SimplePredicate field="TotWinst"
                    operator="greaterOrEqual" value="124.268"/>
                    <False/>
                  </CompoundPredicate>
                </Node>
                <Node id="26" score="50.6528691588" recordCount="107">
                  <CompoundPredicate booleanOperator="surrogate">
                    <SimplePredicate field="TotWinst" operator="lessThan"
                    value="124.268"/>
                    <SimplePredicate field="TotWinst" operator="isMissing"/>
                  </CompoundPredicate>
                </Node>
              </Node>
            </Node>
          </Node>
        </Node>
      </Node>
    </TreeModel>
</PMML>
```

Bibliografie

H.E. Bakker, E.P. van Duin, L. van der Toorn, en A.P. Wennemers, 2001. Het juiste instrumentarium - Resultaten van een marktonderzoek naar data warehouse-oplossingen.

C. Bellomo, 1999. Unleaching the Power of ERP. *DM Review Magazine*, Februari (1999).

B. Benz en J.R. Durant, 2003. *XML Programming Bible*. New York, Wiley Publishing Inc.

A. Berson en S.J. Smith, 1997. *Datawarehousing, Datamining en OLAP*. Schoonhoven, Academic Service.

A. Berson, S. Smith, en K. Thearling, 1999. *Building Data Mining Applications for CRM*. New York, McGraw-Hill.

J. Blomme, 2004. Knowledge Discovery in Data - Bedrijfsintelligentie door Data Mining.

L. Bollen en M. Vluggen, 2001. Strategic Enterprise Management - De Volwassenheidsfase van Enterprise Resource Planning? *Tijdschrift voor Bedrijfsadministratie*, Januari/Februari (2001).

N.J.W. Bothof en B.J. Götte, 1998. *Enterprise Resource Planning als omwenteling - De impact van ERP op organisaties*. Amsterdam/Minneapolis, Giarte Media Group.

A.G. Buchner, M. Baumgarten, M.D. Mulvenna, R. Bohm, en S.S. Anand, 1998. Data Mining and XML - Current and Future Issues.

R.B. Chase, F.R. Jacobs, en N.J. Aquilano, 1998. *Operations Management for Competitive Advantage*. New York, Mc Graw Hill.

T. Davenport, 1998. Putting the enterprise into the Enterprise System. *Harvard Business Review*, Juli/Augustus (1998), 121.

J. van Dijk, 1999. XML Whitepaper.

W. Eckerson, 2000. Analyst Insight - Understanding Packaged Analytic Applications. *DM Review Magazine*, Maart (2000).

H. Eskilsson, C. Nyström, en M. Windler, 2003. ERP System Effects - A Comparison of Theory and Practice.

ETL4ALL, 2005. http://www.etl4all.com.

G. Fernandez, 2003. *Data Mining Using SAS Applications*. Boca Raton, Chapmann and Hall/CRC.

R. Groth, 2000. *Data Mining, Building Competitive Advantage*. New Jersey, Prentice-Hall Inc.

Data Mining Group, 2005. http://www.dmg.org.

G.G. Gable H. Klaus, M. Rosemann, 2000. What is ERP? *Information Systems Frontiers*, 2-2 (2000), 141–162.

N.V. IKAN-Software, 2004. User Guide - ETL4ALL Stand-alone version.

W.H. Inmon, 2002. *Building The Data Warehouse*. New York, John Wiley and Sons, derde druk.

F. Kamst, 2000. Kant-en-klare datamarts. *Database Magazine*, November-7 (2000), 18–22.

F. Kamst en D. Coppens, 1999. De machinekamer van het datawarehouse. *Database Magazine*, Oktober-6 (1999), 20–23.

R. Kimball, L. Reeves, M. Ross, en W. Thornthwaite, 1998. *The Data Warehouse Lifecycle Toolkit*. New York, John Wiley and Sons.

R. Kimball en M. Ross, 2002. *The Data Warehouse Toolkit*. New York, John Wiley and Sons, tweede druk.

Extensible Markup Language, 2005. http://www.xml.org.

K.C. Laudon en J.P. Laudon, 2004. *Management Information Systems - Managing the Digital Firm*. New Jersey, Pearson Prentice Hall, 8e druk.

Lycos, 2005. http://webmaster.lycos.co.uk/glossary/O/.

Ondersteuning van Microsoft Office, 2005. http://office.microsoft.com/nl-nl/assistance/HP010967201043.aspx.

D.L. Moody en M.A.R. Kortink, 2000. From enterprise models to dimensional models - a methodology for data warehouse and data mart design. In *Proceedings of the International Workshop on Design and Management of Data Warehouses*, deel 2.

M. Negnevitsky, 2001. *Artificial Intelligence, a Guide to Intelligent Systems*. Harlow, Pearson Education Limited.

J. A. O'Brien, 1998. *Leerboek ICT-toepassingen, Het bedrijfsleven en het Internet, intra/extranetten en electronic commerce*. Schoonhoven, Academic Service.

M.A. Rashid, L. Hossain, en J.D. Patrick, 2002. *Enterprise Resource Planning - Global Opportunities and Challenges*. Hershey, Idea Group Publishing.

W3C Recommendation, 2005. http://www.w3.org/TR/2000/REC-xml-20001006.

SAS, 2005. http://www.sas.com.

W3 Schools, 2005. http://www.w3schools.com.

M.J. Schroeck, 1999. Insights from the Front Line - Market Intelligent Enterprise. *DM Review Magazine*, Maart (1999).

R. Snijders en V. van Reijswoud, 2000. Hoedt u voor het telefoonstrijkijzer! *Database Magazine*, September-5 (2000), 17–20.

Sage Bob Software, 2005. http://www.sagebobsoftware.com.

Kurt Thearling, 1995. An Introduction to Data Mining - Discovering hidden value in your data warehouse.

Two-Crows-Corporation, 1999. Introduction to Datamining and Knowledge Discovery.

J.A.A. van der Veen en H.S.J. Robben, 1997. Supply Chain Management - Een overzicht. Technical Report 1997-3, Nijenrode University.

D. Wettschereck en S. Muller, 2001. Exchanging Data Mining Models with the Predictive Modelling Markup Language. *IDDM01a*, September (2001), 55–66.

Wikipedia, 2005. http://en.wikipedia.org/wiki/Knowledge.

I.H. Witten en E. Frank, 2000. *Data Mining, Practical Machine Learning Tools and Techniques with Jave Implementations*. San Francisco, Morgan Kaufman Publishers.

T. Witteveen, 2003. Datamining Services - Een andere manier van onderzoeken. *EIM, Onderzoek voor Bedrijf en Beleid*, (2003).

D.C. Yen, D.C. Chou, en J. Chang, 2002. A synergic Analysis for Web-based Enterprise Resource Planning Systems. *Computer Standards and Interfaces*, September-24 (2002), 337–346.