Datawarehouse Ontwerp

Sterschema en ETL schema

Mutaties ten opzichte van vorige versies:

Aanpassing	**Datum**
4.2 vuistregels opsporen feiten	Juli 2008
5.1 voorbeelden bottom-up/top-down weggehaald	Juli 2008
7.5 Testen	Juli 2008
3.2 opnemen datatypes	Feb 2009
4.1 verschillende soorten feiten en voorwaarden aan meetwaarden	Feb 2009
5.6 regels opstellen dimensies	Feb 2009
5.7 veelgemaakte fouten bij dimensies	Feb 2009
6.1 soorten feiten en meetwaarden	Feb 2009
7.2 kwaliteit controle	Feb 2009
Richtlijnen apart benoemen	Nov 2009

Blok	BI-5 Business Intelligence
Datum:	Nov 2009
Auteur:	Jeroen Vuurens

Inhoudsopgave

1 Inleiding

datawarehouse

Een datawarehouse kun je zien als een groot opslagpakhuis waar je alle gegevens die relevant kunnen zijn voor het nemen van bepaalde beslissingen samenbrengt. Die gegevens kunnen afkomstig zijn uit verschillende bronnen.

snel en interactief management-vragen onderzoeken

Datawarehouses zijn een populair instrument om managementvragen te onderzoeken. We kunnen door het gebruik van datawarehouses meer te weten komen, en sneller en betere beslissingen nemen. Twee belangrijke eigenschappen van datawarehouses zijn dan ook:

- Het kunnen onderzoeken van vragen waarvoor gegevens uit verschillende bronnen moeten worden gecombineerd.
- Nieuwe vraagstukken/problemen/kansen snel en interactief signaleren en kunnen onderzoeken.

ontwerp is maatwerk

Een datawarehouse kan worden gevuld met gegevens uit databases, maar ook uit o.a. Excel sheets en platte tekst bestanden, van zowel binnen het bedrijf maar ook data die van buiten het bedrijf komt. Veelgebruikte bronnen voor commerciële organisaties bevatten gegevens die verband houden met orders, klanten, producten, voorraden, personeel, inkoop, marktonderzoek, concurrentie, etc. Stel we maken een datawarehouse voor een ijssalon. Voor een ijssalon kan bijvoorbeeld de vraag belangrijk zijn in hoeverre je op basis van de weersvoorspelling van Erwin Krol (die op Internet staan) kunt bepalen hoeveel ijs je moet maken en hoeveel personeel je in moet roosteren. Het aantal medewerkers dat elke dag werkte staat misschien in een Excel sheet, terwijl de omzetcijfers in een database staan. De gegevens uit deze verschillende bronnen worden gecopieerd naar een datawarehouse en aan elkaar gekoppeld. Zodoende kunnen vragen worden onderzocht waarvoor verschillende bronnen moeten worden gecombineerd. Een datawarehouse is dus geen standaard product, maar iets dat op maat moet worden gemaakt voor het desbetreffende bedrijf. Een goed datawarehouse maken is echt een uitdaging.

enorm groot

Datawarehouses kunnen enorm groot worden. Wall-Mart (de grootste supermarktketen in de USA) is hard bezig om binnen enkele jaren ALLES te gaan registeren met behulp van RFID chips in een datawarehouse. Naar verwachting betekent groeit daardoor het datawarehouse van Wall-Mart per week met enkele Terabytes (TeraByte = 1000GB). Gemiddeld genomen zal een datawarehouse van een doorsnee bedrijf al snel vele Gigabytes omvatten.

ontworpen voor performance

Omdat datawarehouses enorm groot kunnen worden en bovendien snel antwoord op vragen moeten kunnen geven, is een speciale opslagstructuur nodig die gericht is op snelheid. We praten immers over tabellen met soms miljoenen of zelfs miljarden records. Het datawarehouse wordt daarom ontworpen volgens de structuur van een Sterschema [Kimball]. Bovendien kan ervoor worden gekozen om gegevens samen te vatten (aggregeren) zodat het raadplegen sneller gaat. In het volgende plaatje staat een voorproefje van een relationele database die vertaald is naar een sterschema. Je ziet dat er soms een behoorlijke vertaalslag voor nodig is.

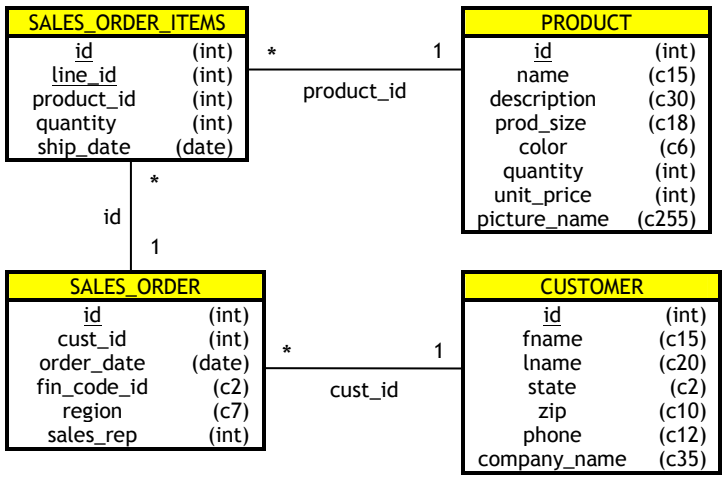


fig. 1.1.1 Operationele Database

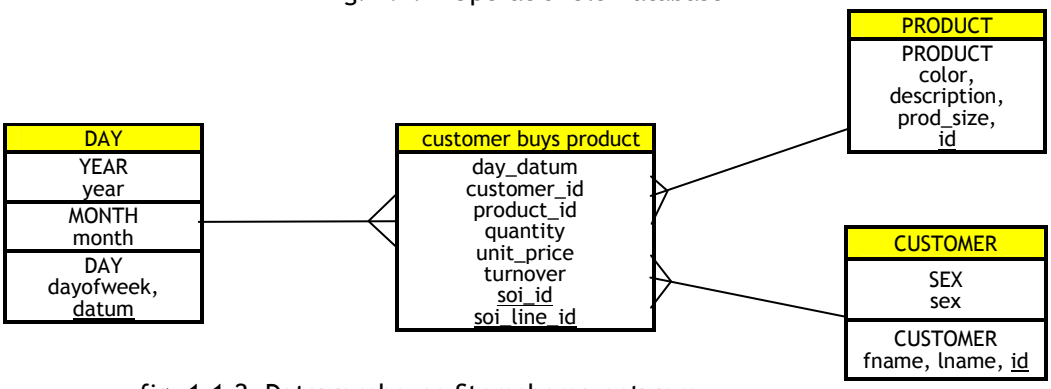


fig. 1.1.2 Datawarehouse Sterschema ontwerp
ps. turnover=omzet

stapsgewijs ontwerpen

Het maken van een vertaling naar een sterschema wordt in stappen aangepakt. De stappen zijn:

1. **bronnenmodel** (bronnen waar het datawarehouse gegevens vandaan haalt)
2. **brongegevensmodel** (klassendiagram van alle brontabellen)
3. **feitmodel** (feiten en welke meetwaarden ze uit welke bron overnemen)
4. **dimensie ontwerp**
5. **datawarehouse ontwerp** (sterschema)
6. **ETL-schema** (specificaties voor het vullen van datawarehouse met data)
7. **datamart** (stukje datawarehouse herontworpen voor bepaald doel)

In deze reader gaan we in hoofdstuk 2 kort in op het in kaart brengen van bronnen. In hoofdstuk 3 gaan we kijken naar feiten en de dimensiematrix. In hoofdstuk 4 gaan we dimensies modelleren. In hoofdstuk 5 komt alles samen in een Sterschema.

ETL

De gegevens uit alle bronnen (waaronder de operationele database) moeten periodiek worden overgezet in het datawarehouse. Aan de hand van de eerder gemaakte diagrammen maken we daarvoor ETL-schema's (Extract Transform Load). Die ETL-schema's kunnen worden ingevoerd in een ETL-tool die het overzetten van gegevens automatiseert. Het maken van ETL-schema's komt in hoofdstuk 6 aan bod.

2 Doel van datawarehouse ontwerp

2.1 Performance

performance bij grote hoeveelheden gegevens

Uitgangspunt bij het ontwerpen van een Operationele Database (en niet bij een datawarehouse) zijn het voorkomen van redundantie (dubbel voorkomen van gegevens), bewaken van integriteit (correctheid) en het verhogen van performance voor opvragen en muteren op recordniveau. Bij een datawarehouse zijn deze zaken onbelangrijk. Hoofddoel is een goede performance bij het onderzoeken van managementvragen die over grote hoeveelheden gegevens worden uitgevoerd. Daar is een heel ander soort ontwerp voor nodig.

sterschema

Het sterschema (fig. 2.1.1) is waarschijnlijk het eenvoudigste en meest gebruikte model om een Datawarehouse te ontwerpen. Een sterschema is een relationeel ontwerp waarbij vanuit één centrale tabel (de feitentabel) verwijzingen staan naar verschillende zogenaamde dimensietabellen.

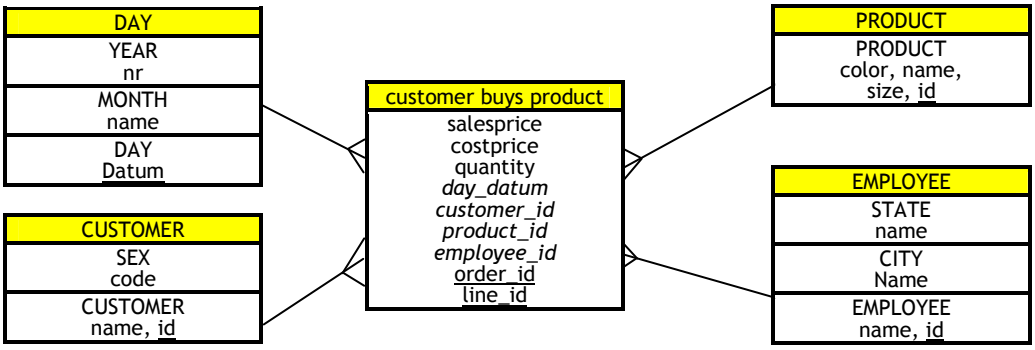


fig. 2.1.1 Sterschema

2.2 Onderzoek mogelijk maken

feitentabel bevat meetwaarden

Door het hoge(re) niveau waarop managementvragen worden onderzocht zijn details (bijvoorbeeld losse verkooprecords) niet interessant, dat zijn er teveel waardoor men geen overzicht meer heeft. Overzicht wordt verkregen door grote hoeveelheden gegevens samen te vatten in bepaalde categorieën, waarbij per categorie de detailgegevens worden *geaggregeerd* (bv. optellen, gemiddelde bepalen, aantal tellen). Bij het onderzoeken van een managementvraag staan daarom bijna altijd meetwaarden centraal. *Meetwaarden* zijn getallen die bij gebeurtenissen horen die je kunt aggregeren. In een datawarehouse komen meetwaarden in een aparte centrale feitentabel. Het overzicht wordt verkregen doordat de verkoopfeiten in wisselende categorieën kunnen worden samengenomen. Gegevens die kunnen worden gebruikt om feiten te categoriseren worden in *dimensietabellen* opgenomen. Doordat de feiten gescheiden van de dimensies worden opgeslagen kunnen we dezelfde feiten gebruiken om vanuit verschillende perspectieven te bekijken (fig. 2.2.1).

aggregeren

meetwaarden

dimensies

	YEAR NR	REGION NAME	QUANTITY SUM
1	1996	Canada	288
2	1996	Central	1176
3	1996	Eastern	1068
4	1996	South	240
5	1996	Western	168
6	1997	Canada	660
7	1997	Central	3216
8	1997	Eastern	3948
9	1997	South	948
10	1997	Western	1068
11	1998	Canada	1260
12	1998	Central	6070
13	1998	Eastern	5907
14	1998	South	1032

Page Items: REGION NAME: Central

	QUANTITY SUM
1996	1176
1997	3216
1998	6070

fig. 2.2.1 Feiten per dimensie

feiten per dimensie categoriseren

De feiten kunnen per verschillende dimensie worden samengenomen (fig. 2.2.1), maar ook in combinaties tussen dimensies . Door dimensies te combineren ontstaan multi-dimensionale overzichten (fig. 2.2.2). Deze kunnen helpen bij het onderzoeken van verbanden tussen dimensies. Bovendien kan door de structuur van de dimensies een bepaalde categorie in meer detail worden bekeken (fig. 2.2.3).

Page Items: Data Point: QUANTITY SUM

	Canada	Central	Eastern	South	Western
1996	288	1176	1068	240	168
1997	660	3216	3948	948	1068
1998	1260	6070	5907	1032	1332
1999	-	26	0	-	-

fig. 2.2.2 Kruistabel van dimensies

Page Items: Data Point: QUANTITY SUM

		Canada	Central	Eastern	South	Western
1996	1	-	144	168	-	-
	2	108	756	456	24	72
	3	144	216	228	84	36
	4	36	60	216	132	60
1997		660	3216	3948	948	1068
1998		1260	6070	5907	1032	1332
1999		-	26	0	-	-

fig. 2.2.3 Interactief meer detail bekijken

2.3 Ad-hoc en interactief

feitentabel klein dimensietabel plat

Gezien de snelheid waarmee men beslissingen wil nemen is snel antwoord op ad-hoc vragen en interactief met de data kunnen ‘stoeien’ belangrijk. De mogelijkheid om de feiten interactief te kunnen bekijken is handig als de performance voldoende is. Het nieuwe overzicht moet er na een paar seconden staan en niet na een paar minuten. De feitentabellen worden daarom in het sterschema zo klein mogelijk gehouden doordat er alleen meetwaarden in staan. De dimensietabellen zijn daarintegen helemaal plat gemaakt, zodat alle relevante gegevens van één feit in één rij staan.

MOLAP vs ROLAP

Uiteindelijk kan een datawarehouse op verschillende manieren worden geimplementeerd. Populair is de MOLAP (Multi-Dimensional Online Analytical Processing) waarbij de data echt multidimensionaal wordt opgeslagen. Tegenhanger daarvan is de ROLAP (Relational Online Analytical Processing). Bij ROLAP komt het datawarehouse in een (gewone) relationele database te staan en de relaties worden dus gelegd mbv Foreign Keys. Voordeel van MOLAP is de snelheid waarmee de feiten per dimensie binnen één feitentabel kunnen worden geagregeerd. ROLAP is doorgaans makkelijker in te voeren (er is al een geschikte relationele database) en goedkoper. Bovendien is ROLAP flexibeler, heeft meer mogelijkheden om meerdere feitentabellen te combineren en uitgebreidere statistische mogelijkheden. Wij gaan bij BI-5 alleen ROLAP gebruiken. Bij ROLAP is het voordeel van een sterschema weliswaar minder groot dan bij MOLAP maar nog steeds enorm ten opzichte van een normale database structuur. Dat komt doordat het aantal rijen dat moet worden gejoined om een overzicht te maken minimaal is.

3 Bronnen

3.1 bronnenmodel

twee invalshoeken

Ontwerp van een datawarehouse begint van twee kanten.

- Wat voor (soort) vragen wil je ermee onderzoeken? (wordt behandeld in de reader Business Intelligence)
- Welke gegevens zijn beschikbaar? (niet alleen intern maar ook van buiten het bedrijf)

goede aanpak essentieel

Een goed datawarehouse ontwerpen is moeilijk. Bovendien resulteren fouten in het datawarehouse ontwerp lang niet altijd in foutmeldingen, maar in foutieve gegevensoverzichten. Dat is funest bij het onderzoeken van managementvragen. Vandaar dat we het datawarehouse goed doordacht en gestructureerd gaan opzetten. We gaan een paar modellen opstellen om stapsgewijs tot een goed ontwerp te komen.

stap 1: bronnentabel

Het eerste model betreft een overzicht van de beschikbare bronnen. Elke tonnetje bevat een IS, database, spreadsheet of ander soort bron. Elke bron komt in een apart tonnetje te staan. In de rechthoeken staan gegevensgebieden. Met behulp van de gegevensgebieden worden aangegeven welke overlap/aanknopingspunten er tussen de bronnen bestaan en wat de kern van de gegevens per bron is. De gegevensgebieden komen niet met één tabel overeen maar met een hele groep tabellen.

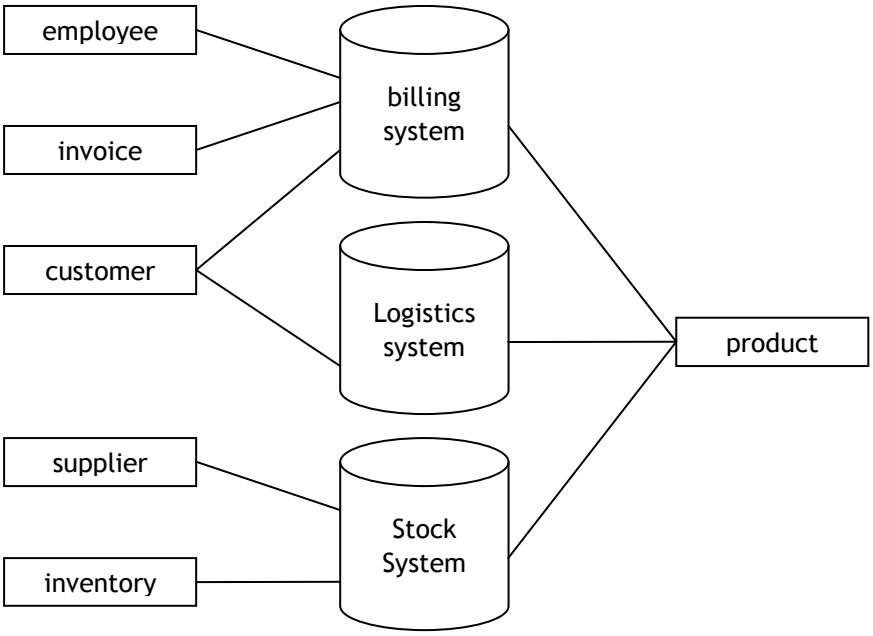


fig. 3.1.1 Bronnenmodel

Richtlijnen:

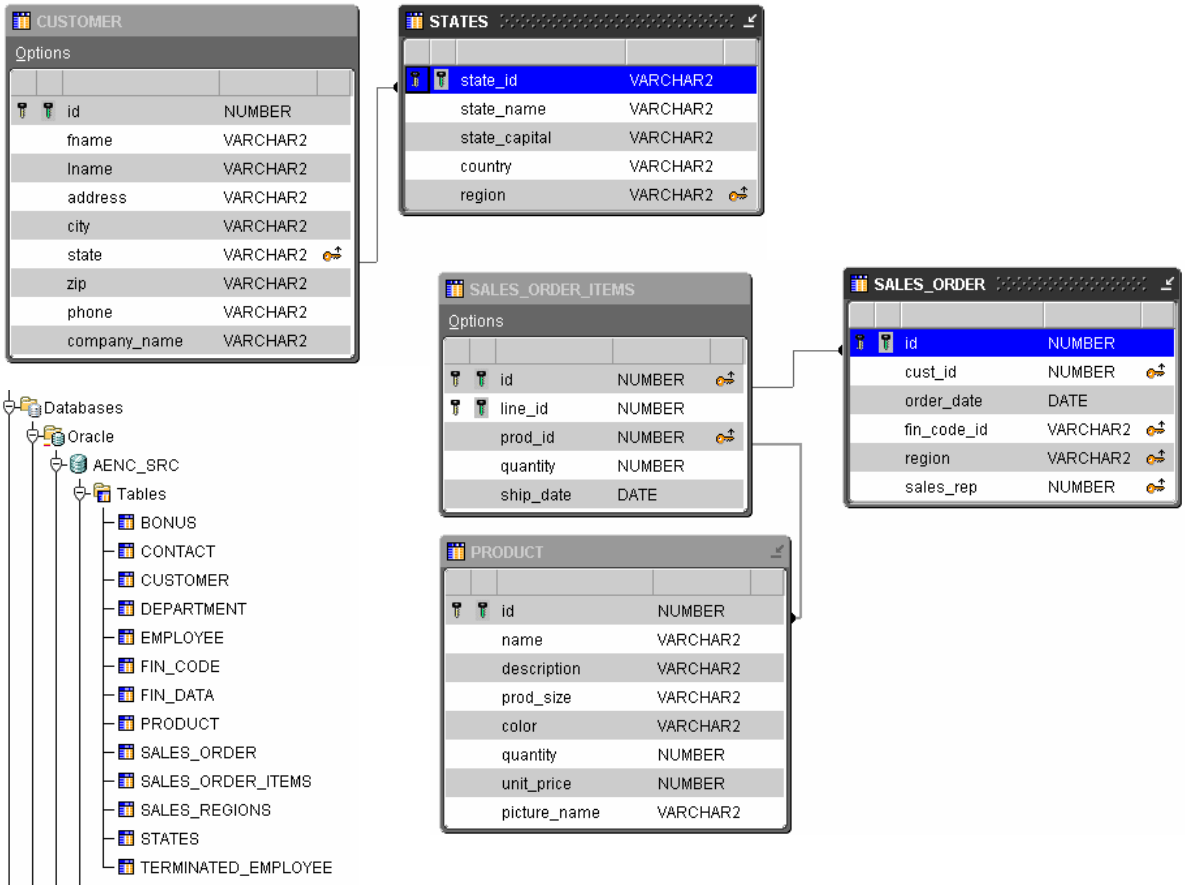
- **Bron**: Elke aparte gegevensbron (informatiesysteem, database, excelsheet, XML-bestand, etc.) wordt in met een apart databasesymbool (tonnetje) weergegeven.
- **Gegevensgebied**: Een gegevensgebied is een samenhangend geheel dat hoort bij een gebeurtenis of object, waarbij het verdelen van de gegevens over verschillende tabellen (bv. product – producttype, order-orderregel) wordt weggelaten. Het is nadrukkelijk niet de bedoeling dat alle tabellen worden genoemd, maar elke tabel moet wel onder een gegevensgebied vallen. Elk gegevensgebied wordt in een rechthoek gezet, met associaties naar de bronnen waarin gegevens uit dit gebied te vinden zijn.
- **Gegevensgebied naam**: Elk gegevensgebied heeft een naam in enkelvoud. Probeer de terminologie van het bedrijf aan te houden en geen nieuwe namen te bedenken om verwarring te voorkomen.

3.2 brongegevensmodel

klassendiagram bronnen

Om wat meer zicht te krijgen op de inhoud en samenhang van de bronnen maken we een relationeel gegevensmodel van de gezamenlijke bronnen. We gebruiken het klassendiagram als techniek om de gegevens te beschrijven. De meeste bronnen zullen de gegevens al in tabellen gestructureerd hebben en kunnen we rechtstreeks overnemen. In dat klassendiagram nemen we bestaande Foreign Keys op en tekenen we de relaties ook als associatie met multipliciteiten.

De tabellen van de AenC database kun je bekijken in Oracle Warehouse Builder. De volgende plaatjes laten een paar tabellen zien:



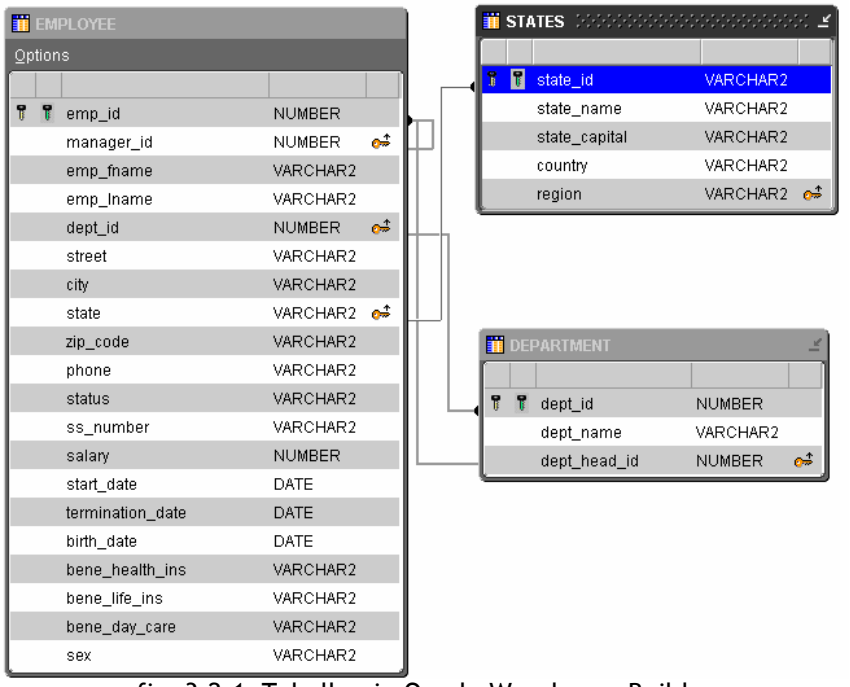
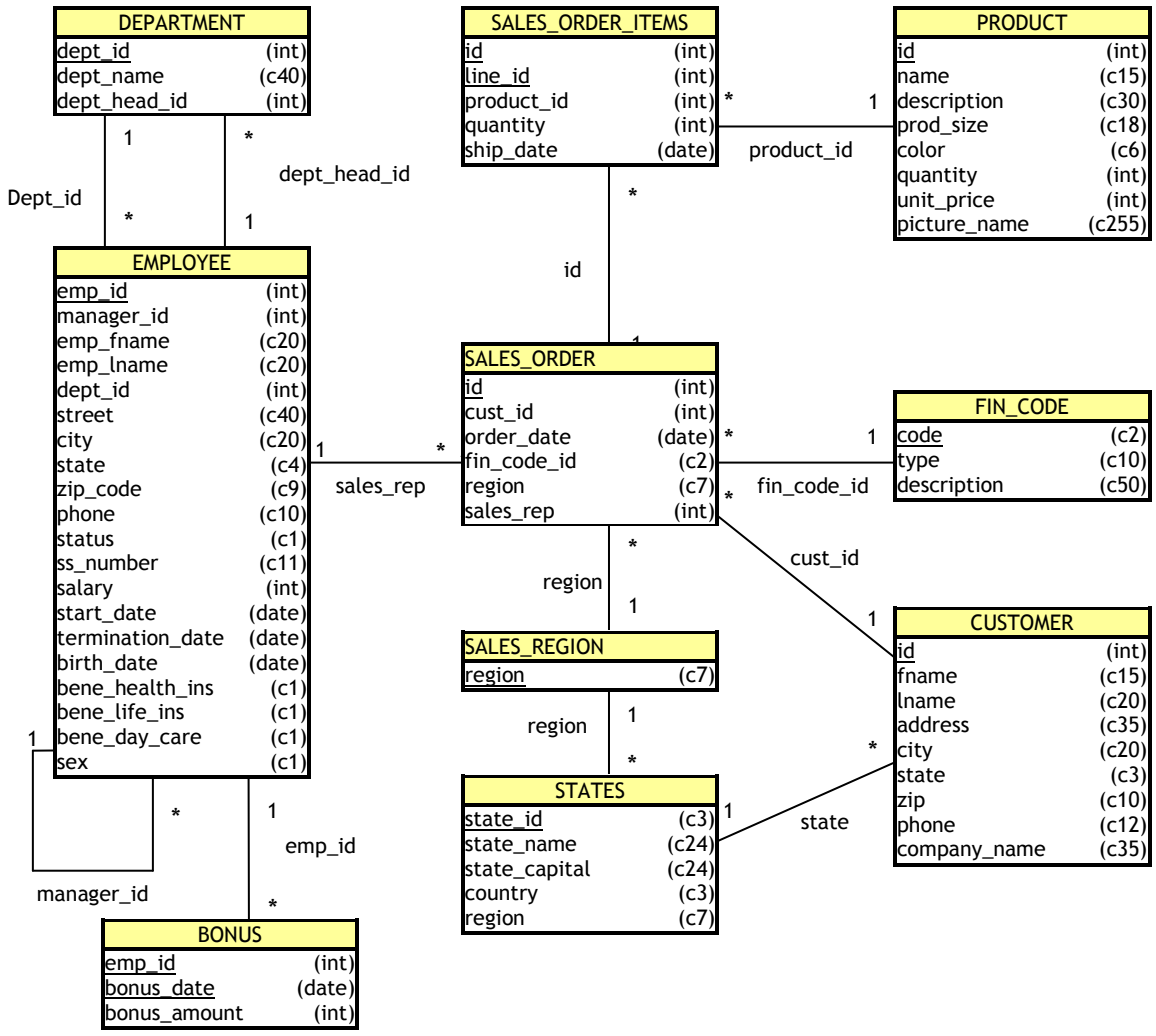


fig. 3.2.1 Tabellen in Oracle Warehouse Builder

Wat bijvoorbeeld nog niet in deze plaatjes staat is dat de tabel SALES_ORDER een CUST_ID heeft en dus eigenlijk naar de tabel CUSTOMER verwijst. In de tabel SALES_ORDER zit ook de kolom SALES_REP die naar EMPLOYEE verwijst. In het brongegevensmodel maken we alle verbanden tussen de tabellen zichtbaar mbv associaties.



Data Dictionary

Department: Een afdeling binnen het bedrijf met een department_head waar employees onder vallen.
Employee: Een medewerker binnen het bedrijf die werkt bij een bepaald department onder een department_head.
Bonus: Een extra bonus in euro's (bonus_amount) die aan een employee is uitgekeerd.
Product: Een beschrijving van een standaard product dat wordt verkocht aan Customers door employees van dit bedrijf.
Sales_order: Een verkoopopdracht die is geplaatst door een customer bij een bepaalde employee. Elke sales_order kan verschillende producten omvatten, die in de bijbehorende sales_order_items tabel staan.
Sales_region: Een opdeling van het verkoopgebied in regio's waarmee eenduidig geregistreerd kan worden in welke region customers wonen en in welke region verkopen worden gedaan.
States: Een staat in de USA of Canada waar customers en employees wonen.
Customer: De contactpersoon van een bedrijf dat van A&C die producten koopt.
Fin_code: De financiële code waaronder een sales_order in de boekhouding geregistreerd wordt.

fig. 3.2.2 brongegevensmodel

stap 2: brongegevensmodel

We gebruiken een brongegevensmodel (0) als basis om het datawarehouse te ontwerpen. Voor dat doel is het handig als de Foreign Keys en Primary Keys er duidelijk in staan (NB het is dubbelop en volgens database-experts vast niet correct, maar wel erg praktisch om de Foreign Key te gebruiken als naam van de associatie). Het bij de hand hebben van alle datatypes voorkomt eveneens later veel speurwerk.

één brongegevens-model

Soms put een datawarehouse gegevens uit één database, maar in de praktijk worden vaak verschillende bronnen/databases in een datawarehouse samengebracht. Indien de brontabellen over verschillende databases verdeeld zijn maken we er toch liefst één brongegevensmodel zodat duidelijk wordt hoe de bronnen gekoppeld zijn.

Informatie kwaliteit

Indien dezelfde tabel of hetzelfde attribuut in meer dan één bron voorkomt, is het nuttig om de inhoud te bekijken. Het kan zijn dat er op een andere manier waarden aan het attribuut worden gegeven (geslacht M/V of 0/1). Het kan ook zijn dat codes niet overeenkomen (landcode 23: Canada vs. landcode 23: Australië). We spreken dan van gebrekkige informatiekwaliteit, die moet worden opgelost (bv bij het ETL-proces) om tot een correct gevuld datawarehouse te komen.

datatypes opnemen

Het opnemen van datatypes is handig omdat datatypes in het datawarehouse ontwerp overeen moeten komen, of er bewust een transformatie moet worden gemaakt in het ETL-proces. Indien dit niet het geval is, loopt men het risico dat sommige data niet in het datawarehouse gezet kan worden of verminkt raakt. In beide gevallen zijn dat fouten die moeilijk te achterhalen zijn en beter voorkomen kunnen worden.

Richtlijnen:

Geïntegreerd gegevensmodel: Maak één brongegevensmodel waarin de gegevens van alle gegevensbronnen zijn opgenomen en waarin duidelijk wordt hoe de gegevens uit verschillende bronnen gekoppeld zijn.

Tabel: Neem een tabel op voor elke afzonderlijke tabel die in een gegevensbron voorkomt. Een tabel beschrijft de gegevens per record, waarbij elk record slechts één waarde voor elk attribuut mag bevatten. Elk record beschrijft één object, gebeurtenis of eigenschappen daarvan.

Naamgeving: Elke tabel en attribuut heeft een naam in enkelvoud. Probeer de terminologie van het bedrijf aan te houden en geen nieuwe namen te bedenken om verwarring te voorkomen.

Datatypes: Bij elk attribuut dient het exacte datatype van elk attribuut te worden opgenomen zodat bij het datawarehouse- en ETL-ontwerp de juiste datatypen en benodigde transformaties kunnen worden gemodelleerd.

Primary Key: In elk tabel wordt de primaire sleutel onderstreept: een (combinatie van) attribuut(en) die uniek is voor elk record.

Associaties: Neem een associatie op indien het mogelijk is om een record in de ene tabel te herleiden tot precies één record in een andere tabel. In principe levert een relationele database altijd 1-* relaties (de primary key is de 1-kant, de foreign key is de *-kant). Bij de associatie wordt de koppeling beschreven door bijvoorbeeld de foreign key te noemen.

Data dictionary: In een data dictionary worden alle begrippen (tabel- en attribuutnamen) toegelicht zodat helder is wat de betekenis van de gegevens is.

4 Feiten

4.1 Meetwaarden en dimensiegegevens

feit

Binnen datawarehouses maakt men onderscheid tussen feiten en dimensies. Met een feit bedoelt hij een gebeurtenis of toestand.

- **Gebeurtenis**: bv. iemand koopt iets, iemand brengt een product terug, een student legt een tentamen af tentamen. Gebeurtenissen kun je over het algemeen tellen (het aantal keer dat een klant iets koopt, het aantal keer dat een student een tentamen maakt, etc.).
- **Toestand/momentopname**: bv. de snelheid van een auto, de hoeveelheid geld in kas, het aantal klanten in de winkel. Een feit is een voortschrijdende toestand die op bepaalde momenten wordt gemeten.

De attributen die aan feiten te relateren zijn worden gesplitst in twee groepen.

meetwaarde

- **Meetwaarden**: zijn numeriek, bij elke gebeurtenis hoort per meetwaarde maar één waarde, en het is zinvol om er (ter analyse) berekeningen op los te laten (fig. 4.1.1). N.B. Datum en tijd zijn dus geen meetwaarden.
  Elke gebeurtenis heeft precies één waarde per meetwaarde en elke meetwaarde moet passen bij één enkele gebeurtenis. Als een gebeurtenis bijvoorbeeld het produceren van één product is, kan daar niet de bezettingsgraad van de machine bij worden opgenomen als meetwaarde omdat die alleen berekend kan worden als de productietijd van alle producten worden bekeken.

Feit	Meetwaarde	zinvolle berekening
Customer buys product	• quantity • price • discount	Sum(quantityl) Sum(quantity*price) Sum(quantity*discount)
Employee receives bonus	• amount	Sum(amount) Max(amount)

Student maakt tentamen	• Behaalde cijfer • Benodigde tijd	Avg(cijfer)

fig. 4.1.1 Meetwaarden met voorbeelden van zinvolle berekeningen

dimensiegegeven

- **Dimensiegegevens**: Een gegeven kan alleen een dimensiegegevens zijn als er bij alle gebeurtenissen maar één dimensiewaarde hoort. Dimensies worden gebruikt om de feiten te categoriseren (fig. 4.1.2). Het zijn over het algemeen attributen die iets permanents van een object/categorie beschrijven die aan een gebeurtenis gerelateerd is.

Feit	Mogelijke dimensiegegeven
Customer buys product	• customer: sex, city, date of birth • Product: color, weight, shiptime, type • Time: day of week, month, year • Branch: surface in m2, city, country • Employee: name, years in service, …
Employee receives bonus	• Employee: name, years in service, … • Time: year

Student maakt tentamen	• Student: naam, major_opleiding, leeftijd, studeert_sinds • Module: #studiepunten, opleiding, vakgebied, moeilijkheidsgraad, heeft_proeftentamen • Tentamen: aantal_vragen, is_herkansing, multiple_choice • Tijdstip: starttijd, dag van de week, weeknr, blok, jaar

fig. 4.1.2 Dimensies bij feiten

dimensiegegeven = categorie voor feiten

De hoeveelheid dimensiegegevens kan erg omvangrijk worden. Als een customer iets koopt gaat het niet alleen om wie de customer is en wat het product is, maar ook de plek waar de customer het koopt, het moment waarop de customer het koopt, etc. **Dimensiegegevens worden gebruikt om feiten te kunnen categoriseren**. Zo kan men de totale omzet bepalen per geslacht, woonplaats, leeftijd, kleur, gewicht, levertijd, etc.

gegeven dat zowel bij feit als dimensie kan horen

Sommige dimensiegegevens zijn ook als meetwaarde te gebruiken. Leeftijd is bij klant koopt product een dimensiegegeven. Het zegt namelijk iets over de klant (een dimensie) en niet zozeer over de gebeurtenis. Willen we van een gebeurtenissen de gemiddelde leeftijd weten dan kan het echter toch handig zijn om deze als meetwaarde op te nemen. Als dimensiewaarde kan het handig zijn om de feiten per leeftijdscategorie samen te nemen (kopen oude mensen meer dan jonge mensen). Als leeftijd op beide manieren moet kunnen worden gebruikt dan kan dat alleen als het gegeven als meetwaarde én als dimensiegegeven wordt opgenomen.

4.2 feitmodel

stap 3: feitmodel

Aan de hand van het doel van Business Intelligence en het brongegevensmodel worden de feiten en meetwaarden (fig. 4.2.1) in kaart gebracht. De feiten worden geformuleerd als relevante gebeurtenis die uit een bron is af te leiden. Indien mogelijk worden daar (*afgeleide*) meetwaarden bij genoemd, met de tabel waar die meetwaarde uit komt. Bij de brontabellen wordt de brondatabase genoemd.

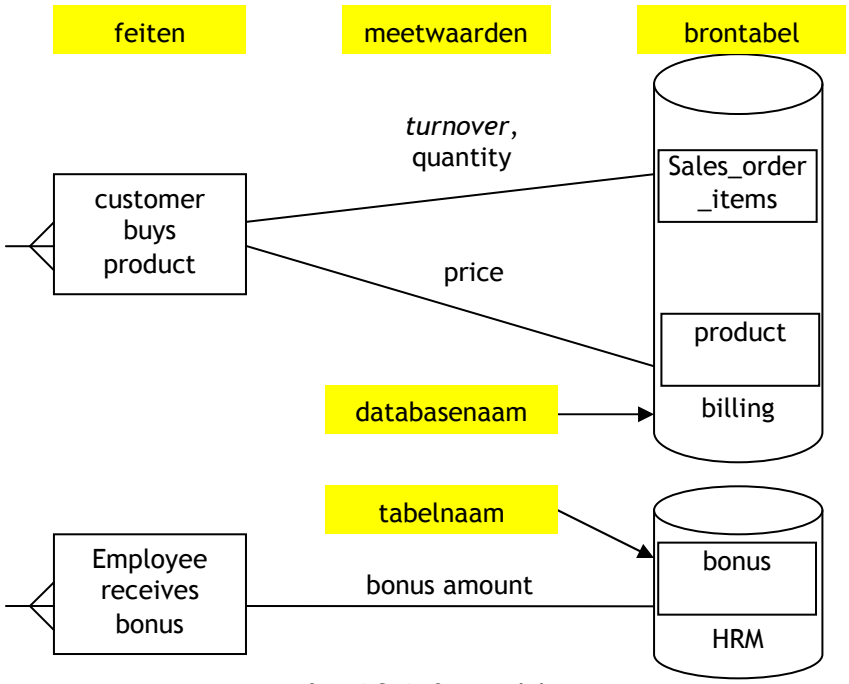


fig. 4.2.1 feitmodel

Voorbeeld van brontabellen:
PRODUCT

id	name	description	prod_size	color	quantity	price
101	t-shirt	v-neck	L	green	1	$10.00
102	t-shirt	v-neck	XL	blue	1	$10.00
201	socks		8-9	white	3	$ 2.00

SALES ORDER ITEMS

id	line_id	product_id	quantity	ship_date
1	1	101	10	1-1-2006
1	2	201	5	1-1-2006
2	1	101	3	10-1-2006

vuistregels opsporen feiten

Een feit is altijd een gebeurtenis of momentopname. Met een paar eenvoudige vuistregels kun je feiten opsporen in een brongegevensmodel:

foreign Keys

Bevat Foreign Keys. Een kandidaat voor een feitentabel bevat over het algemeen meer dan één Foreign Key terwijl er geen Foreign Key vanuit een potentiële dimensie naar de feitentabel kan bestaan. Volgens deze regel zijn bonus en sales_order_items de twee belangrijkste kandidaten.

Twee gebeurtenissen kunnen ook naar elkaar verwijzen, denk aan *kopen* en *retourneren* of *betalen*. Dit zijn aparte gebeurtenissen omdat ze op een ander moment plaats kunnen vinden. Het kan ook zijn dat verschillende gebeurtenissen in één tabel staan, indien een betaling plaatsvindt wordt bijvoorbeeld de betaaldatum ingevuld die daarvoor NULL was. Deze verschillende gebeurtenissen kunnen we zien als verschillende feiten.

tijdsaspect

Bevat een tijdsaspect. Bij een gebeurtenis of momentopname moet altijd een tijdstip horen. Dat tijdstip kan eventueel in een andere tabel staan (zoals in het geval van sales_order_items de datum in sales_order staat). Maar ook als het tijdstip niet geregistreerd wordt (als bijvoorbeeld de retourdatum niet in de tabel retour staat), maar het duidelijk is dat er wel een tijdstip bestaat, is waarschijnlijk sprake van een gebeurtenis.

Soms kiest men ervoor om een gebeurtenis te implementeren met meer dan één tabel, zoals *sales_order_items* en *sales_order*. Beiden tabellen refereren naar dezelfde gebeurtenis (vinden tegelijk plaats). Het verdient de voorkeur atomaire feiten te modelleren, d.w.z. op het laagste detailniveau. De tabel waar de foreign key uit de onderlinge relatie zit, geeft de gebeurtenis op atomair niveau weer.

koppeling records en gebeurtenissen

Koppeling tussen records en gebeurtenissen. Een brontabel is alleen geschikt als bron voor een feitentabel als je de frequentie van gebeurtenissen kunt achterhalen. Elke gebeurtenis wordt beschreven door precies 1 record in de tabel. Elke keer als een medewerker een bonus ontvangt moet er dus een nieuw record in die tabel bijkomen. Vanuit dat record kunnen we via Foreign Keys bijbehorende gegevens uit andere tabellen erbij zoeken, zoals naam van de medewerker en naam van het departement.

zinvolle berekening/ meetwaarde

Zinvolle berekeningen. Bij elke relevante gebeurtenis hoort één of meer zinvolle berekeningen. In het slechte geval is dat zonder meetwaarden, maar door te kijken naar het aantal gebeurtenissen (bijvoorbeeld het aantal verkeersovertredingen). Maar meestal zijn dat berekeningen over **meetwaarden** zoals het berekenen van de omzet, gemiddelde bonus per medewerker of de winst per productlijn.

onveranderlijk

Gebeurtenissen zijn onveranderlijk. Een gebeurtenis uit het verleden kan niet worden gewijzigd, hoogstens gecorrigeerd door een nieuwe gebeurtenis. Als een klant twee dagen na aankoop een product terugbrengt, dan wordt de aankoop niet verwijderd, maar de retournering geregistreerd. De verkoopgebeurtenis heeft immers wel degelijk plaatsgevonden. Dit in tegenstelling tot dimensies, die door de tijd heen wel langzaam kunnen veranderen. Klanten worden ouder, verhuizen, krijgen een andere baan, een hoger opleidingsniveau. Van producten wordt de prijs of verpakking soms aangepast.

relevant

Relevantie. Het enkel toevoegen van een record in een tabel maakt nog geen gebeurtenis. Denk als voorbeeld aan een product tabel. Bij een product kan een introductiedatum, een prijs en een Foreign Key naar een productgroep horen. Ook al is aan een hoop voorwaarden voldaan, toch dient het toevoegen van een product aan het assortiment niet snel als gebeurtenis te worden gemodelleerd. Dat heeft te maken met relevantie. Als er geen zinvolle berekeningen denkbaar zijn, of er geen verband tussen het moment van de gebeurtenis en succes kan worden gelegd, dan betreft het geen relevante feitentabel. Het kan dan misschien wel als dimensie bij andere feitentabellen worden gemodelleerd.

Richtlijnen:
Feit: een gebeurtenis die op een bepaald tijdstip (bv. een verkoop) of over een bepaald tijdsinterval (bv. een autorit) plaatsvindt.
Feitnaam: De gebeurtenis van het feit wordt over het algemeen in een werkwoord verwoord met daarbij de 2 meest kenmerkende dimensies als onderwerp/leidend voorwerp.
Brontabel: Elke gebeurtenis wordt gekoppeld aan brontabellen waaruit kan worden opgemaakt dat een gebeurtenis heeft plaatsgevonden en welke meetwaarden erbij horen.
Meetwaarde: Indien mogelijk wordt bij de koppeling tussen een feit en een brontabel aangegeven welke meetwaarden en afgeleide meetwaarden daaruit kunnen worden gehaald. Een meetwaarde moet aggregeerbaar zijn en mag slechts één waarde per feit hebben.
Gegevensbron: Om de tabellen wordt en grote databaseton getekend met daarin de naam van de gegevensbron waar de tabellen zich in bevinden.

5 Dimensies

5.1 Inventariseren

dimensie = categorie voor feiten

De context van de feiten wordt beschreven met behulp van dimensiegegevens. Deze dimensiegegevens gebruiken we niet om mee te rekenen maar om de feiten in categorieën in te delen. Dimensies zijn over het algemeen objectverzamelingen die in relatie tot de gebeurtenis staan. Alle dimensiegegevens komen in een dimensie terecht. Bij het feit 'klant koopt product' horen bijvoorbeeld de dimensies tijdstip, verkoper, klant en product. Maar daarnaast zijn misschien ook andere dimensies zinvol zoals filiaal, verpakking, verkoopkanaal, betaalmiddel, etc.

richtlijnen dimensies

Dimensies moeten voldoen aan de volgende richtlijnen:

- Bij elke gebeurtenis hoort maar één dimensiewaarde
- De dimensie is onafhankelijk van gebeurtenissen beschreven
- Dimensies bieden zinvolle categorieën (dimensiewaarden) om gebeurtenissen in op te delen

Het inventariseren van dimensies kan vanuit twee invalshoeken.

- **bottom-up**: Alle beschikbare attributen inventariseren (brongegevensmodel) waarvan bij elke gebeurtenis slechts één waarde hoort. De attributen die in de feitentabel komen of die geen nuttige categorie opleveren schrappen. Vervolgens vanuit de resterende attributen komen tot objecten die nuttig zijn als dimensie.
- **top-down**: Vanuit beleid, doelstellingen, managementvragen komen tot zinvolle categorieën. Bij die categorieën probeer je dan relevante attributen te zoeken.

bottom-up focust op gegevens die er zijn top-down focust op gegevens die je wilt

Zowel de bottom-up als de top-down benaderingen bieden voordelen. De bottom-up benadering is snel te realiseren want alle gegevens zijn voorhanden. Met de top-down benadering kijk je meer naar waar je behoefte aan hebt. Kijken we naar de tijd dimensie van het feit 'klant koopt product' dan zien we dat de top-down benadering laat zien dat het bedrijf veel meer interesse heeft om de resultaten naar (boekhoudkundige) kwartalen in te delen. Bij departement zien we dat er wensen zijn om de resultaten van grote en kleine departementen met elkaar te vergelijken. Dit zijn wensen die gemakkelijk zijn te realiseren en waar je met de bottom-up benadering misschien te snel aan voorbij was gegaan.

Sommige gegevens in de top-down lijst staan niet in het brongegevensmodel, zoals verkoper en filiaal. In sommige gevallen is er wel aan die gegevens te komen (als het niet in een database staat, staat het wel in een of ander Excel bestandje). Indien de gegevens niet voorhanden zijn kan men besluiten om de betreffende gegevens vanaf nu wel te registreren of misschien kopen.

inventarisatie dimensie-gegevens-tabel

We kunnen ten behoeve van de later te maken ETL-schema's de inventarisatie van dimensiegegevens uitbreiden met de brontabel, zoals in fig. 5.1.1 . Maar het hoofddoel van deze inventarisatie is om tot goede dimensies te komen voor ons datawarehouse ontwerp. De vervolgstap is om daar goede dimensieontwerpen bij te maken.

Feit	Dimensie	Gegeven	Bron
customer buys product	Department	name size	department employee
	Employee	name sex years in service	employee
	Time	datum quarter season year	sales_order
	product	size color	product
	Customer	city state	customer
Employee receives bonus	Time	year	bonus
	Department	name size	department employee
	Employee	name sex years in service salary schale	employee

fig. 5.1.1 inventarisatie dimensiegegevens

5.2 Relatie tussen Feit en Dimensie

alleen associaties van feit naar dimensie

In een sterschema is maar één soort associatie tussen tabellen mogelijk: van een feitentabel naar een dimensietabel. **Bij elk feit hoort precies één dimensiewaarde**, omgekeerd mag bij een dimensiewaarde wel meer dan één feit horen. Dimensies kunnen dus geen directe relaties hebben met andere dimensies. We modelleren in principe altijd relationeel, m.a.w. de feitentabel krijgt Foreign Keys die naar de Primary Keys van de dimensies verwijzen.

primaire sleutel

Elke dimensie heeft een primaire sleutel nodig, een unieke identificatie die maar bij één object/record van de dimensietabel hoort. Denk aan ‘klantnummer’ voor klant, ‘kenteken’ voor vervoersmiddel, ‘sofi-nummer’ voor Nederlanders, ‘studentnummer’ voor studenten, maar ook ‘datum’ als primaire sleutel voor de dimensie DAG.

primaire sleutel bestaat uit één integer attribuut

surrogaatsleutel

Een geschikte primaire sleutel voor een dimensietabel bestaat liefst uit één numeriek attribuut (datum is ook numeriek). Daardoor kan de feitentabel zo klein mogelijk blijven (die zal namelijk Foreign Keys naar de dimensies bevatten) en kan een relationeel datawarehouse feitentabellen en dimensietabellen sneller joinen. Dit zorgt voor de veel betere performance waar we naar streven. De primaire sleutels voor dimensietabellen kunnen soms uit de bron worden overgenomen, maar indien er geen goede primaire sleutel voorhanden is kan men er ook voor kiezen in het datawarehouse een kunstmatige sleutel te creëren. We spreken dan van een surrogaatsleutel.

datawarehouse ontwerp moet ook voor de toekomst correct zijn

Datawarehouses moeten de ‘tand des tijds’ kunnen doorstaan. Hou er altijd rekening mee dat in de toekomst nieuwe data wordt toegevoegd. Indien we in de huidige situatie telefoonnummer als Primary Key van Klant dienst kan doen (omdat er op dit moment bij elk telefoonnummer maar één klant hoort) dan moet je je nog steeds afvragen of dat in de toekomst ook zo zal blijven. Als in de toekomst mogelijk een telefoonnummer voorkomt waar verschillende klanten en dus verschillende klantnamen bij horen, dan is telefoonnummer dus geen goede Primary Key voor Klant!

5.3 Dimensiehiërarchie

hiërarchieniveau

In veel gevallen kun je een dimensie zien als een object, zoals klant, verkoper of product. Zulke dimensies zijn zinvol om gebeurtenissen per object samen te nemen (verkoop per klant, verkoper of product). De dimensiegegevens kunnen echter binnen de dimensie ook weer categorieën vormen (klanten per woonplaats, verkopers per geslacht, producten per kleur). Van deze dimensiegegevens kunnen we aparte hiërarchieniveaus maken. Categorieën binnen een dimensie.

hiërarchie

Tussen verschillende hiërarchieniveaus kan een één op veel verband bestaan. Bij een tijdhiërarchie kunnen we dagen bijvoorbeeld indelen in maanden, maanden weer indelen in kwartalen en kwartalen indelen in jaren. Zodoende kunnen we categorieën ordenen tot een hiërarchie. Een hiërarchie zegt alleen iets over de relatie tussen twee aangrenzende niveaus (een kwartaal bestaat uit maanden, een maand bestaat uit dagen). Twee hiërarchieniveaus zijn correct op elkaar gestapeld als elk object van het onderliggende niveau binnen precies één categorie van het bovenliggende niveau voorkomt. Bij elke datum hoort één maand en alle maanden tezamen bevatten alle dagen. Elke maand hoort weer bij precies één kwartaal, dus de gezamenlijke kwartalen omvatten alle mogelijke maanden.

fig. 5.3.1 Mogelijke hiërarchie van een tijdsdimensie

hiërarchie bestaat uit hiërarchielevels met een zinvolle 1-N relatie

Een hiërarchie tekenen we altijd van boven naar beneden. In een hiërarchie bestaat alleen een relatie tussen twee aangrenzende niveaus. Elk bovenliggend niveau moet een zinvolle 1 op veel relatie hebben met het onderliggende niveau. De relatie tussen twee verder uit elkaar gelegen niveaus (hier dag en jaar) bestaat alleen indirect via de tussenliggende niveaus.

Bij elk object hoort maar één bovenliggende categorie

De eis dat in een hiërarchie elke object op het onderste niveau maar behoort tot één categorie op een hoger niveau is keihard! Elke dag kan dus maar tot één maand behoren. Twee niveaus mogen alleen boven elkaar staan als aan die eis is voldaan.

In figuur 5.3.1 hoort elke datum bij één maand, terwijl een maand bestaat uit 28-31 dagen. De maand die hoort bij deze specifieke datum hoort ook maar bij één jaar (want het gaat niet om een jaarloze maand januari, maar bijvoorbeeld de maand januari die hoort bij de datums 1-1-2003 t/m 31-1-2003). 1 jaar bevat 12 maanden.

feitabel wordt altijd gekoppeld aan het laagste dimensieniveau

In een sterschema mag je feitentabellen alleen koppelen aan het laagste niveau van dimensies. Feitentabellen die we aan bovenstaande hiërarchie koppelen moeten dus als Foreign Key op het niveau van dag hebben. Het laagste niveau van de hiërarchie bevat dus altijd de Primary Key van de dimensietabel.

dimensies op hetzelfde niveau sluiten elkaar uit en omvatten gezamenlijk alle feiten

Het indelen in hiërarchieën geeft ons de mogelijkheid om categorieën te maken op verschillende niveaus. Die dimensiewaarden op hetzelfde niveau:

- sluiten elkaar altijd uit (jaren hebben geen overlap; er bestaat geen dag die bij meerdere jaren hoort)
- dekken de hele dimensie af (de gezamenlijke jaren omvatten alle bestaande dagen).

Deze regels zijn noodzakelijk om te garanderen dat alle soorten overzichten correct zijn. Als je de omzetten van alle dagen bekijkt en die bij elkaar optelt, kom je op hetzelfde totaal als dat je de omzetten van alle jaren bij elkaar optelt.

levelattributen

Elk hiërarchielevel kan 1 of meer levelattributen bevatten. Een hiërarchielevel ‘klant’ kan bijvoorbeeld naast een klantnaam ook een sofi-nummer, geboortedatum, geboorteplaats, etc. hebben. Merk op dat het soms een keuze is of je de attributen op één level plaatst of over levels verdeeld. Indien je geïnteresseerd bent om feiten te bekijken per geboorteplaats van de klant kun je die op een ander hiërarchielevel zetten.

5.4 Dimensie ontwerp

dimensietabel in een Sterschema

In een sterschema tekenen we per dimensie een tabel . De hiërarchieniveaus worden duidelijk aangegeven met elk een apart vak en een naam in hoofdletters. Bij een hiërarchieniveau kunnen verschillende attributen horen, die worden eronder gezet.

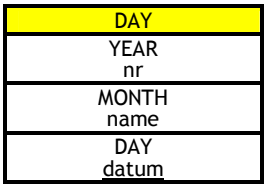


fig. 5.4.1 DAY dimensie

naamgeving dimensie

De dimensie moet ook een naam hebben. In de meeste boeken wordt bovenstaande dimensie tijd genoemd. Om verwarring te voorkomen kan het ook handig zijn om de naam van het laagste niveau over te nemen.

niet unieke hiërarchieniveaus

Sommige hierrachieniveaus beschikken niet over unieke attributen. Dat is bijvoorbeeld het geval bij maanden (want de maand februari komt in verschillende jaren voor) en bij woonplaatsen (want steden als Moskou en St. Petersburg komen in verschillende landen voor). Dit zijn wel degelijk echte hiërarchieën. Want het plaatsje Moskou in Nederland is toch echt een andere stad dan Moskou in Rusland, en de resultaten van deze twee mogen niet zomaar worden samengenomen. Alleen is het bovenliggende niveau (jaar of land) nodig om elk plaatsje uniek te identificeren.

drill-down

Straks als het datawarehouse gebouwd is en we gaan managementvragen onderzoeken, dan kunnen we de feiten op elk gewenst niveau bekijken (jaar, maand, dag). Door de hiërarchische structuur kunnen we bovendien op hoog niveau beginnen (bv. jaar) en dan inzoomen op het jaar dat ons het meest interessant lijkt. Op de maand(en) die ons het meest interessant lijkt kunnen we dan weer verder inzoomen. We noemen dit drillen of drill-downs (wordt behandeld in de OLAP tutorial) en het is een van de meest gebruikte interactieve technieken om vragen te onderzoeken terwijl je het overzicht behoudt.

hiërarchieniveau moet overzicht nuttig verhogen

Het indelen in hiërarchische niveaus moet nuttig zijn bij de analyse. Een voorbeeld van een (waarschijnlijk) niet zinnig hiërarchieniveau is bijvoorbeeld achternaam van een klant. Bij de achternaam ‘Jansen’ kunnen weliswaar meer klanten horen, maar hoe nuttig kan het zijn om verkoopfeiten op achternaam samen te nemen? Bovendien zijn er tal van achternamen waar maar één klant bij hoort, m.a.w. het aantal items wordt nauwelijks gereduceerd als je van het niveau klant naar het niveau achternaam gaat.

vuistregel: streef naar een 1-10 relatie

Uiteindelijk moet de hiërarchische opbouw van dimensies bruikbaar zijn bij analyse. Streef als vuistregel naar een afname van het aantal niveaus met een factor 10. Drillen op een item levert dan gemiddeld 10 onderliggende items op. Zodat bijvoorbeeld drillen op een land niet meteen 1000 klanten van dat land laat zien, maar eerst de 13 provincies en daaronder misschien de steden. Er zijn uitzonderingen op deze vuistregel, zoals een opdeling van personen naar geslacht. Bij dat soort hiërarchieën kun je per geslacht resultaten bekijken en is de analist meestal niet geïnteresseerd in het naar beneden drillen.

dimensie met één attribuut

Er kunnen ook dimensies zijn die uit één niveau bestaan en zelfs maar uit één attribuut (fig. 5.4.2). In theorie is het zelfs mogelijk om van elk dimensiegegeven een aparte dimensie te maken. Zo is het niet verboden om van de woonplaats van een klant een aparte dimensie te maken. Maar het moet natuurlijk wel zinnig zijn. Indien je klant als dimensie opneemt is het logisch om de woonplaats in de hiërarchie van klant op te nemen. Dat biedt immers de meeste mogelijkheden met de analyse en de feitentabel blijft kleiner waardoor de performance beter is. Streef er dus doorgaans naar om hiërarchieën aan te brengen waar mogelijk.

fig. 5.4.2 Dimensie met één attribuut

5.5 Alternatieve hiërarchieën

Bij het opzetten van hiërarchieën zijn er soms verschillende mogelijkheden:

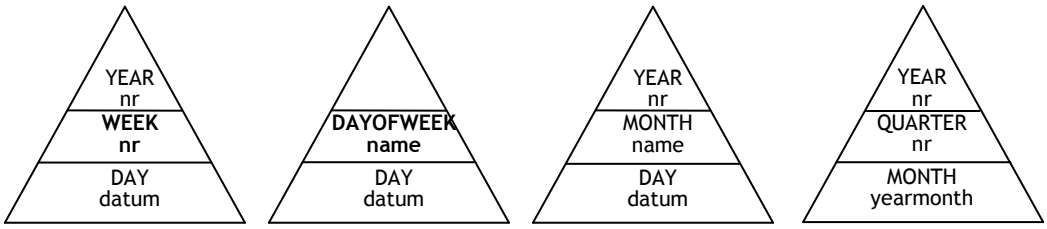


fig. 5.5.1 Alternatieve hiërarchieën

Week is redelijkerwijs niet in een maand onder te brengen. Want een week kan voor de ene helft in de ene maand vallen en voor de andere helft in de volgende maand. Op een soortgelijke manier kan een week ook eigenlijk niet onder een jaar vallen, maar we kunnen het oplossen door een goede afspraak daarover te maken (week 1 van de kalender is de eerste week van het nieuwe jaar). Het is overigens meestal geen goede keuze om week in een tijddimensie op te nemen.

hiërarchieën samennemen

Bij dagnaam kunnen we de keuze maken om die onder te brengen in de rechter hiërarchie. Elke datum hoort bij één dagnaam. Bv. 28-8-2006 hoort bij maandag, die desbetreffende maandag hoort bij augustus, die maand augustus hoort bij 2006. We kunnen dagnaam dus desgewenst in de hiërarchie van JAAR->MAAND->DAGNAAM->DAG plaatsen. Merk wel op dat deze manier van modelleren implementatie afhankelijk is. Indien men de dagnaam alleen gebruikt om de omzet tussen dagen over alle jaren heen te vergelijken kan men beter kiezen voor een alternatieve (aparte) hiërarchie.

analyse mogelijkheden nemen toe door meer niveaus en alternatieve hiërarchieën

Van de *alternatieve* hiërarchieën (fig. 5.5.1) kunnen we het best één dimensie maken. We kunnen weliswaar bij de analyse waarschijnlijk maar één hiërarchie tegelijk gebruiken, maar we kunnen wel steeds kiezen welke hiërarchie dat is. Het aantal mogelijkheden voor analyse neemt dus toe als we de dimensies uitbreiden met alternatieve hiërarchieën en meer hiërarchische niveaus.

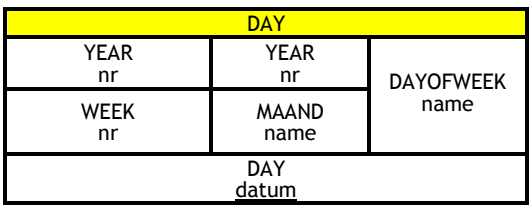


fig. 5.5.2 Dimensie met 3 alternatieve hiërarchieën

regels alternatieve hiërarchieën

Er zijn een paar regels voor alternatieve hiërarchieën.

- De hiërarchieën hebben altijd één gemeenschappelijk laagste niveau, in dit geval is dat DAG. Verschillende hiërarchieën mogen overigens wel meer gemeenschappelijke lagere niveaus hebben.
- Elke hiërarchie is een apart pad naar boven, waarbij er geen gemeenschappelijke niveaus meer bestaan. In dit geval is JAAR daarom gedupliceerd. Merk bovendien op dat JAAR en JAAR niet hetzelfde hoeven te zijn, aangezien 1-1-2005 onder week 52 2004 kan vallen.
- Het laagste gemeenschappelijke niveau bevat de voor de hele dimensietabel unieke Primary Key. Feitentabellen die naar deze dimensie verwijzen krijgen dus als Foreign Key een kopie van die Primary Key waarde (zoals altijd bij relationele databases)

5.6 Dimensieontwerp

Stap 4: dimensie ontwerp

De afzonderlijke dimensies worden eerst los ontwerpen in een diagram met de naam dimensieontwerp.

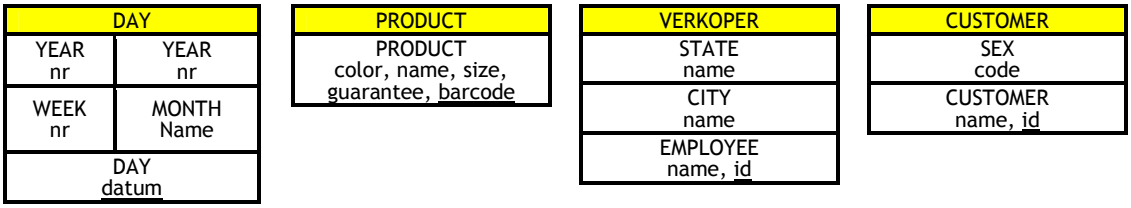


fig. 5.6.1 Dimensieontwerp

Regels opstellen dimensies

Richtlijnen:

- **Dimensie**: Een dimensie moet geschikt zijn om gebeurtenissen te categoriseren en om meetwaarden per categoriewaarde te aggregeren. Vaak is een dimensie een eigenschap die verschillende gebeurtenissen gemeenschappelijk hebben (bv. orderstatus) of een klasse waarvan elke gebeurtenis bij één object hoort.
- **Laagste niveau**: Elke dimensie heeft maar één laagste niveau. Elke gebeurtenis uit een gerelateerde feitentabel kan maar gekoppeld worden aan één waarde van elke dimensie. De naam van het laagste dimensieniveau is gelijk aan de naam van de dimensie.
- **Unieke dimensieobjecten**: Elk dimensieobject komt in principe maar één keer in een dimensie voor, met uitzondering van verschillende versies van een dimensie waarbij eigenschappen zijn gewijzigd (bv. een auto die en andere kleur heeft gekregen). Elke dimensie heeft een onderstreepte primary key op het laagste niveau die bestaat uit één numeriek attribuut.
- **Hiërarchie**: Dimensieobjecten worden in een hiërarchie van eigenschappen geplaatst waardoor zij op verschillende manieren in categorieën kunnen worden ingedeeld. In een hiërarchie worden attributen in verschillende niveaus boven elkaar geplaatst (bv land boven woonplaats) als de harde eis geldt dat alle dimensiewaarden die dezelfde eigenschap op een lager niveau hebben ook dezelfde eigenschap op het bovenliggende niveau hebben (bv alle klanten uit Amsterdam komen ook uit Nederland).
- **Hiërarchieniveau**: Elk hiërarchieniveau heeft een duidelijk apart vak, met een naam in hoofdletters, en daaronder één of meer attributen die bij dat niveau horen.
- **Alternatieve hiërarchie**: Twee verschillende hiërarchieën kunnen zonder problemen naast elkaar bestaan. De enige eisen zijn dat het laagste niveau van een dimensie altijd wordt gedeeld door alle hiërarchieën (er is dus maar één onderste hiërarchieniveau in elke dimensie) en twee alternatieve hiërarchieën kunnen op hoger niveau niet samenkomen in een gemeenschappelijk niveau.

5.7 Veelgemaakte fouten

Bij het ontwerpen van dimensies kunnen fouten ervoor zorgen dat het datawarehouse niet gebouwd kan worden of onvoorspelbare (corrupte) uitkomsten geeft.

KLANT	KLANT	ORDER	DAY	
WOONPLAATS naam	LAND naam	LAND naam	YEAR nr	
GESLACHT code	POSTCODE code	DATUM datum	WEEK nr	MONTH Name
KLANT nr	KLANT nr	ORDER nr	DAY datum	

fig. 5.7.1 Veelgemaakte dimensiefouten

Tussen dimensieniveaus moet een 1 op veel relatie liggen. Dat betekent in de dimensie links bv. dat alle klanten die van het vrouwelijk geslacht zijn, in dezelfde woonplaats moeten wonen. Als dat niet overeenkomt met de huidige situatie, maar ook als het in de toekomst voor zou kunnen komen dat er een vrouw in een andere stad woont, dan is deze hiërarchie niet correct. Merk op dat het omdraaien van woonplaats en geslacht ook niet goed is. Dan zouden alle klanten uit dezelfde woonplaats allemaal hetzelfde geslacht moeten hebben.

Het tweede voorbeeld is iets lastiger. Stel dat we in de data ook klanten tegen kunnen komen uit een land waar ze helemaal geen postcode hebben, dan zou voor die mensen de postcode een NULL-waarde krijgen. NULL-waarden in dimensies zijn uit den boze, die dien je ten allen tijde te vermijden. Maar ook de hiërarchie klopt dan niet meer, want alle klanten met een NULL-waarde als postcode moeten dan uit hetzelfde land komen, wat onwaarschijnlijk is. Op het hoogste niveau in een dimensiehiërarchie wordt soms wel een attribuut toegepast wat een NULL-waarde kan hebben, maar dan dient men de NULL-waarde te vervangen door een niet NULL-waarde (bv. ‘onbekend’).

In het derde voorbeeld is de dimensie eigenlijk geen dimensie maar een gebeurtenis. Indien er sprake is van toegevoegde waarde, dan zou men daar beter een feitentabel van kunnen maken. Maar in de meeste gevallen is die toegevoegde waarde er niet. Een datawarehouse wordt niet gebruikt door de operationele processen om een enkele order op te zoeken, maar om totalen te berekenen. Een belangrijke reden om het niet te doen is performance. Niet alleen wordt de feitentabel uitgebreid met een foreign key, maar een join tussen 2 feitentabellen die een grote hoeveelheid records bevatten, is moordend voor de performance. De bovenliggende niveaus datum en land hebben een onderlinge veel op veel relatie en kunnen dus nooit in dezelfde hiërarchie staan.

In het laatste voorbeeld is er gezondigd tegen de regel dat alternatieve hiërarchieën geen gemeenschappelijk hoger niveau mogen hebben. Het probleem wat zich hier voordoet is dat langs via maand de datum 31-12-2007 in het jaar 2007 valt, en via week dezelfde datum in 2008 zou kunnen vallen. Dat kan niet. In theorie leidt dat tot onvoorspelbare resultaten, omdat de dag bij de totaaltelling van beide jaren meegeteld zou kunnen worden.

6 Sterschema

6.1 Feitentabel

Atomaire gebeurtenis

Feiten zijn meestal atomaire gebeurtenissen. Met atomair bedoelen we dat de gebeurtenis niet op te delen is naar kleinere gebeurtenissen zonder verlies van informatie. Op hoger niveau spreken we van samengestelde gebeurtenissen, zoals het doen van boodschappen. Elk gekocht artikel (waarbij 2x hetzelfde artikel wel wordt samengenomen) kan dan worden gezien als een atomair feit.

momentopname

Er bestaan ook andere vormen van feiten, zoals het meten van een toestand op een bepaald moment in de tijd. Denk daarbij aan veranderlijke eigenschappen zoals snelheid, temperatuur, gewicht, hoeveelheid geld in kas. In plaats van een gebeurtenis gaat het hier meer om een continu proces of een gebeurtenis over langere tijd (een autorit, een studie).

verschillende gebeurtenissen leveren aparte feitentabellen

De vuistregel voor het maken van feitentabellen is: verschillende gebeurtenissen leveren elk een aparte feitentabel op. Dat zorgt ervoor dat het datawarehouse zo veel mogelijk details over de gebeurtenissen kan bevatten. Elk feit uit het feitmodel levert dus een aparte feitentabel op. Van elke feitentabel kun je een apart sterschema tekenen, maar dan moeten dezelfde dimensies wel consistent met elkaar zijn. Men kan ook één sterschema maken met verschillende feitentabellen daarin.

soorten attributen in een feitentabel

Een feitentabel kan maar 3 soorten attributen bevatten:

- **Meetwaarden**: numerieke attributen waarop een zinvolle berekening kan worden uitgevoerd.
- **Foreign keys**: verwijzingen naar dimensietabellen, waarbij de voorkeur uitgaat naar een numerieke sleutel van 1 attribuut.
- **Identificerende attributen**: attributen waarmee de data naar de bron te herleiden zijn. Deze attributen hebben geen functie in het datawarehouse, maar zijn nodig om een ETL-proces te kunnen inrichten om het datawarehouse uit te breiden.

6.2 Relaties tekenen

feitentabel

We ontwerpen een datawarehouse aan de hand van een sterschema. Uit de feit/gegevens/bron-tabel wordt elk feit een aparte feitentabel. In een sterschema tekenen we die centraal in het midden. De dimensies die erbij horen tekenen we er omheen.

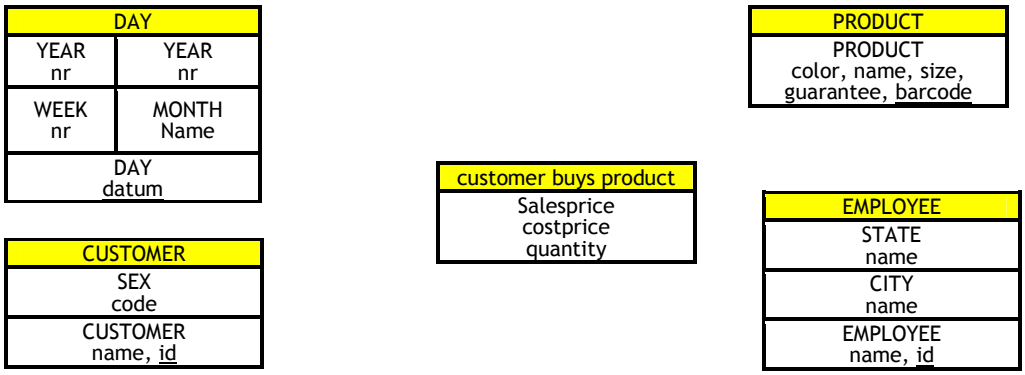


fig. 6.2.1 Feitentabel met dimensies

1 op veel relaties tussen feitentabel en dimensietabellen

Er is slechts één soort relatie in een sterschema mogelijk, en dat is een veel op 1 relatie tussen feitentabel en (het laagste niveau van) een dimensietabel. Die wordt getekend als een lijn met een harkje aan de veel-kant. Bij elk verkoopfeit hoort precies één klant, bij elke klant kunnen nul of meer verkoopfeiten horen.

primary Key dimensies en foreign keys feitentabel

Naast het visuele harkje wordt de relatie relationeel gelegd. Elke dimensie heeft daarvoor een primary key op het laagste niveau nodig die we onderstrepen. De feitentabel bevat Foreign Keys naar de dimensies.

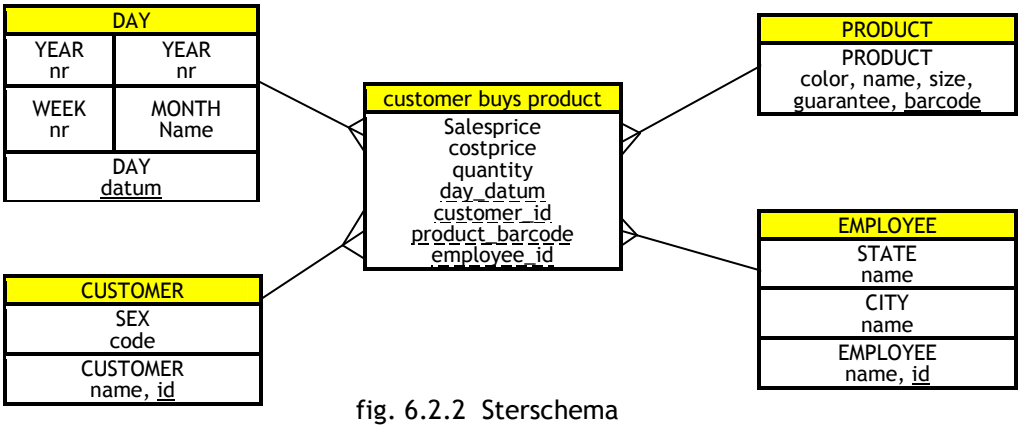


fig. 6.2.2 Sterschema

stap 5: sterschema

Het sterschema is het uiteindelijke ontwerp van het datawarehouse. De volgende paragrafen bespreken nog enkele overweging die verwerkt moeten worden.

6.3 Datawarehouse en Datamarts

Datawarehouse: zoveel mogelijk details behouden

Vaak wordt in de praktijk een opdeling gemaakt tussen een datawarehouse en datamarts. In een datawarehouse streeft men over het algemeen naar het behoud van zoveel mogelijk detail. Zelfs als dat in de huidige situatie niet nodig is, wil men de mogelijkheden voor analyse in de toekomst zoveel mogelijk open houden (think big).

Datamart: ingericht voor een specifiek analyse doel

Een datamart is een stukje datawarehouse dat is ingericht voor een specifiek (analyse-)doel. Door de opdeling tussen datawarehouses en datamarts kan het datawarehouse perfect worden gemodelleerd als historisch allesomvattend archief. De performanceproblemen bij analyse worden opgelost in de datamart. Deze hoeft niet alle data te bevatten, kan geaggregeerd zijn en feitentabellen met samengestelde feiten bevatten.

scheiden archief van analyseplatform

Het fysieke onderscheid tussen een datawarehouse en een datamart is soms moeilijk vast te stellen. Kimball ziet een datawarehouse als een verzameling van datamarts terwijl Inmon een datamart ziet als een afgeleide van een datawarehouse. De voordelen van Inmons visie is dat we voor het datawarehouse indien we daar geen rechtstreekse analyse op doen een andere (misschien meer geschikte) opslagstructuur kunnen kiezen. Bovendien kunnen de datamarts op verschillende servers worden ondergebracht wat ook weer performancevoordeel kan opleveren. In praktijk wordt Inmon dan ook vaker gevolgd op dit punt.

6.4 Atomic en Aggregate feitentabellen

atomic aggregate

Los van het type feitentabel kun je kiezen voor een Atomic feitentabel of een Aggregate feitentabel. In een Atomic feitentabel is elk feit ondeelbaar en een apart record. In een Aggregate feitentabel is elk feit samengesteld uit (vaak een optelling van) verschillende atomic feiten. Atomic feitentabellen zijn kleiner en hebben dus een betere performance bij analyse.

laagste graan bestaat uit atomaire gebeurtenissen

Het datawarehouse bestaat doorgaans uit Atomic feitentabellen. Een atomic feitentabel bevat data op het laagste, uitgebreidste en meest gedetailleerde niveau: elke atomaire (ondeelbare) gebeurtenis is een apart feit/record. Bij elke gebeurtenis mag maar één dimensiewaarde horen. Elke gebeurtenis dat 1 klant 1product op 1 plaats bij 1 verkoper in 1 filiaal via 1 verkoopkanaal, etc, etc, heeft gekocht is dus één apart feit. Indien een klant op een bepaald moment 10 verschillende producten tegelijkertijd zien we dat als 10 atomaire aankopen die tegelijkertijd plaats hebben gevonden. Ook indien dezelfde gebeurtenis meerdere keren voorkomt zijn dat aparte feiten. Het modelleren van atomaire feiten is het meest gedetailleerde niveau en we noemen we het laagste granulariteitsniveau (grofheid) of fijnste 'graan'. Meetwaarden kunnen dan bijvoorbeeld zijn de prijs van het product en het aantal van dat ene product dat die klant op dat moment, etc, etc, heeft gekocht.

aggregeren

Verbetering van performance van datamarts wordt verkregen door de atomaire feiten uit het datawarehouse te aggregeren. Dat kan door gebeurtenissen met dezelfde dimensiewaarden samen te nemen. Bestelt iemand 2 x een cola met dezelfde dimensiewaarden (datum, verkoper, etc.), dan wordt dat één feit met als meetwaarde het aantal 2.

aggregeren tot hoger graan

De performancewinst wordt nog groter als we de dimensies van een aggregatie naar een hoger niveau te brengen. Modelleren we in plaats van product - productgroep, dan worden bijvoorbeeld alle verschillende drankjes uit het restaurantvoorbeeld samengenomen. Indien we bovendien maand als laagste niveau de tijdsdimensie nemen, dan worden alle drankjes van één klant bij één verkoper over de hele maand samengenomen. Bij het feit hoort dan niet meer één productprijs maar een totaalbedrag (een optelling van het aantal per product * prijs per product). We kunnen dan wel per maand het aantal drankjes zien, maar niet meer op welke dag en wat voor drankje het was. We noemen dat ook wel het verhogen van de graan van de feitentabel. De feitentabel wordt een grovere samenvatting van losse feiten, waardoor detail verloren gaat, maar de feitentabel kleiner wordt (en dus de performance beter).

overbodige dimensies schrappen

Weer een andere variant van een aggregatie is het schrappen van dimensies waarin men niet of zelden geïnteresseerd is. Indien de verkoper wordt geschrapt worden bestellingen van hetzelfde product door dezelfde klant maar bij verschillende verkopers samengenomen.

meetwaarden op hoger graan

Atomaire meetwaarden uit het datawarehouse worden samengenomen tot geaggregeerde meetwaarden in een datamart (zoals totale_aantal, totaalbedrag en gemiddelde_snelheid). Doorgaans neemt de hoeveelheid detail naar een hoger graanniveau dus af. Bij enkele uitzonderingen komt er op een hoger graanniveau nieuwe meetwaarden bij, zoals op het niveau van order er sprake kan zijn van een kortingsbedrag, dat niet voor individuele orderregels niet geldt.

afweging tussen detail en performance

Het voordeel van een datawarehouse op het laagste 'graan' is dat zoveel mogelijk detail behouden blijft. Dat biedt meer mogelijkheden met de analyse. Het voordeel van een hoger graan in de daaruit afgeleide datamarts is dat de feitentabel kleiner wordt en dus de performance beter. Indien voor een bepaalde doel bepaalde dimensies niet nodig zijn kan het dus een overweging zijn om die weg te laten.

aggregate: combinatie van dimensies uniek

Het meest gebruikelijk is om bij het aggregeren een (minimale) combinatie van dimensies uniek te maken. In andere woorden, twee gebeurtenissen met dezelfde dimensiewaarden worden samengenomen tot één feit. In het voorbeeld van fig. 6.2.2 , indien men een restaurant modelleert en een klant plaatst op een avond 20 verschillende bestellingen voor hetzelfde drankje bij dezelfde serveerster, dan wordt die 20 bestelrecords in het bronbestand één feit in de aggregate feitentabel. Het verlies informatie is dat uit het datawarehouse niet meer af te leiden is of iemand 20x 1 drankje heeft besteld of 1x 20 drankjes. Het voordeel is dat 20 records zijn samengenomen tot één.

6.5 Conformed Dimensions

gemeenschappelijke dimensies

Een datawarehouse met meer dan één feitentabel heeft gemeenschappelijke dimensies nodig om die feiten met elkaar te kunnen combineren.

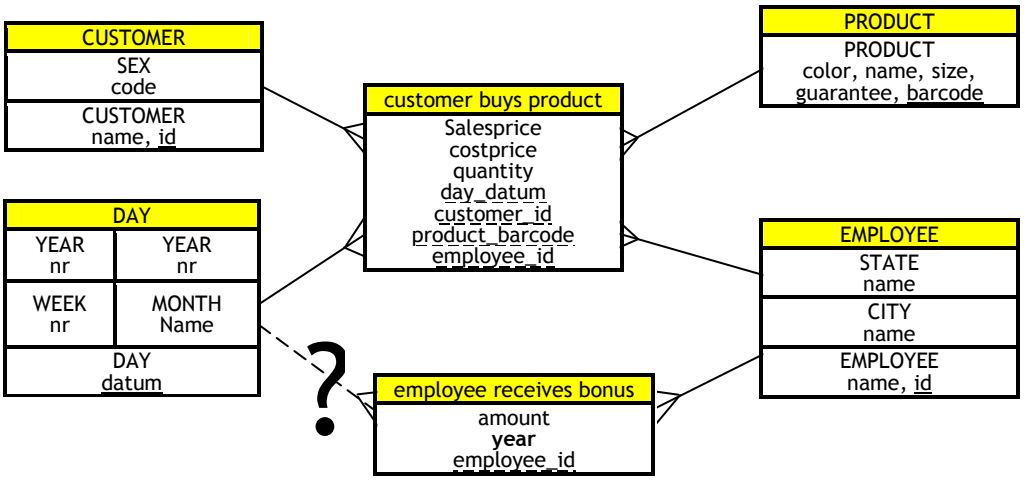


fig. 6.5.1 Twee feitentabellen

feiten naar een hoger dimensieniveau brengen

Indien er twee feitentabellen zijn die gebruik maken van dezelfde dimensie(s) dan teken je de dimensies maar één keer. De dimensies zijn echter niet altijd op hetzelfde hiërarchieniveau beschikbaar voor alle feitentabellen. In fig. 6.5.1 is te zien dat bonussen worden bijgehouden per jaar en verkopen per dag. Indien men deze feiten in combinatie met elkaar wil onderzoeken over de tijd, dan moeten ze aan dezelfde tijddimensie hangen. En feitentabellen kunnen alleen worden gekoppeld aan het laagste niveau van een dimensie. We kunnen uit een jaartal geen datum afleiden. Dan is de enige oplossing om de datums van verkoop naar jaarniveau te brengen.

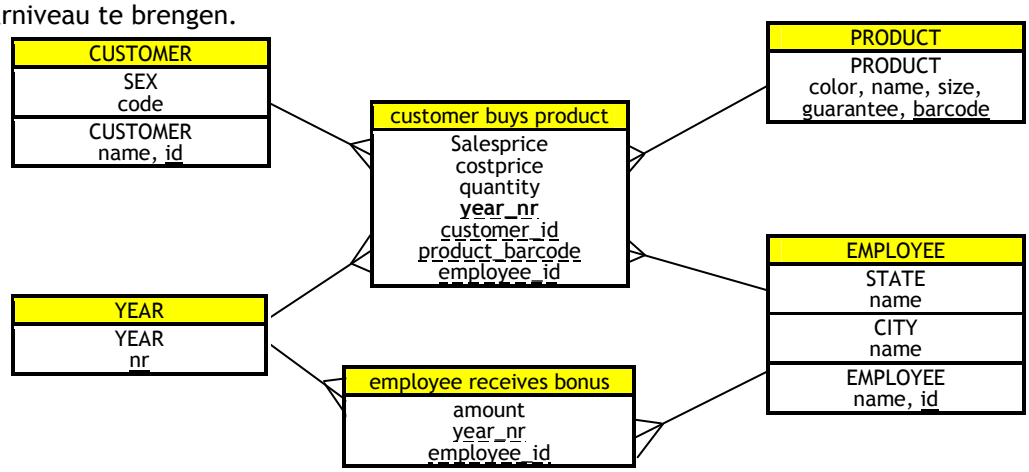


fig. 6.5.2 Conformed dimensions

conformed dimensions

Nu is aan de eis voldaan dat de feitentabellen aan het laagste niveau van de dimensies zijn gekoppeld. JAAR en VERKOPER zijn conformed dimensions die we kunnen gebruiken om verkoopfeiten en bonusfeiten te combineren in overzichten.

oplossingen tegen verlies van detail

De consequentie van bovenstaande oplossing is dat verkoopfeiten nu niet meer op maand kunnen worden bekeken. Indien dit toch gewenst is kan men naast een JAAR dimensie een aparte DAG dimensie opnemen en VERKOOP aan beiden koppelen. In bovenstaand geval kiest men er ook wel voor om het moment van bonusuitreiking op een vaste dag (bv 1 januari of 31 december) te stellen om zodoende toch een DAG dimensie te kunnen handhaven. Beide oplossingen hebben hun nadelen.

6.6 Identificerende attributen feitentabel

Herhaalbare ETL-schema's

In tegenstelling tot een dimensie heeft een feitentabel geen primary key nodig. De tabel wordt niet gebruikt voor mutaties en er zijn ook geen andere tabellen die naar feitentabellen verwijzen. Wel heeft een feitentabel identificerende attributen nodig om de brondata te kunnen herleiden. Zodoende kan tijdens het ETL proces worden gezien of een gebeurtenis in de brondata al in het datawarehouse is openomen. Het ETL-proces kan daarmee herhaalbaar worden gemaakt.

atomic feitentabel: degenerate dimension

In de meeste gevallen is elke gebeurtenis te herleiden tot één record in een brontabel. In die brontabel staat het aantal records dus gelijk aan het aantal gebeurtenissen. Zo'n tabel noemen we de key-source-table. Uit de key-source-table levert elk record 1 gebeurtenis op in een atomic feitentabel. Men dient de primary key van de key-source-table over te nemen in de feitentabel om het ETL-proces herhaalbaar te kunnen maken. Kimball noemt dit identificerende attribuut een 'degenerate dimension'. In zijn optiek creëer je eigenlijk een dimensie met ordernummers of kassabonnummers. Omdat het niet zinnig is om een feitentabel te joinen met een verder lege dimensietabel laat je die dimensietabel eigenlijk weg maar laat je de foreign key (die dan geen foreign key meer is) staan. Dit is dan geen verwijzing naar een dimensie meer, maar kan wel als identificerend attribuut bij ETL worden gebruikt.

aggregate: minimale unieke combinatie foreign keys

Bij een aggregate worden verschillende atomic feiten gecombineerd tot één geaggregeerde gebeurtenis. Er dient dan geaggregeerd te worden over de combinatie van alle dimensiesleutels. Een aggregate krijgt als identificerende attributen een zo klein mogelijke combinatie van dimensiesleutels die uniek is. Soms is dat een combinatie van alle foreign keys maar soms is een bepaalde dimensie niet nodig om een unieke sleutel op te leveren.

Richtlijnen:

Naam feitentabel: Een feitentabel stelt een gebeurtenis voor en niet records uit één brontabel. Het verdient voorkeur om als naam de gebeurtenis te verwerken in de terminologie van de brontabellen.

Type feitentabel: Er moet een duidelijke keuze worden gemaakt tussen een atomic feitentabel en een geaggregeerde feitentabel.

Attributen: Bijeen feitentabel streeft men naar zo klein mogelijke records. Daarom komen in een feitentabel slechts 3 soorten attributen voor: meetwaarden, foreign keys naar dimensies en identificerende attributen. Er horen geen andere attributen dan deze 3 soorten in te staan.

Meetwaarde: Elke meetwaarde die in een feitentabel staat moet geschikt zijn om zinvol te aggregeren (optellen, gemiddelde bepalen, etc.). Het resultaat moet een zinvolle uitkomst zijn. Men kan zonder meetwaarden altijd het aantal gebeurtenissen tellen, daar neemt men geen meetwaarde voor op. Zo bestaan er feitentabellen zonder meetwaarden.

Foreign Keys: Een feitentabel heeft een foreign key naar elke direct daaraan verbonden dimensie. Dit is een kopie van de primaire sleutel van die dimensie. Dat kan alleen als aan de eis is voldaan dat bij elke gebeurtenis precies één dimensiewaarde hoort.

Identificerende attributen: Elke gebeurtenis moet herleidbaar zijn tot de brongegevens zodat mutaties (bv een orderwijziging) kunnen worden overgenomen in het datawarehouse. Daarom moet elke atomic feitentabel een kopie bevatten van de primaire sleutel van de key-source-table en een geaggregeerde feitentabel de minimale combinatie van foreign keys die uniek is als identificerende attributen bevatten.

7 ETL

7.1 Mappings

verschillende bronnen

Nuttige gegevens komen in de praktijk lang niet altijd uit één database. Verschillende afdelingen of informatiesystemen houden gegevens op hun eigen manier bij. Tussen die verschillende gegevensbronnen zitten redundante (overtollige, dubbele) gegevens, maar ook tabellen die elkaar aanvullen. HRM zal bijvoorbeeld andere data van verkopers bijhouden dan een verkoopafdeling. Sommige gegevens zitten niet eens in een database, denk aan de excel-sheetjes die Managers er soms op nahouden. Alle gegevens die we willen betrekken bij analyse zullen we moeten samenbrengen in één datawarehouse. Met een ETL-tool kan men de data uit verschillende bronnen halen, en die verwerken/samenbrengen in één datawarehouse.

ETL

Het ETL-proces omvat 3 fases:

- **Extract**: data uit de bron ophalen en indien nodig in een gestructureerd formaat brengen (van attributen met datatypen) om verder te kunnen verwerken.
- **Transform**: verwerken van de data tot meetwaarden of dimensiegegevens. Onder deze stap valt o.a.
  - het vertalen van data (bijvoorbeeld Male->M Female->F),
  - consolideren van data (Male, Mr., M, sex: 1 moeten allemaal worden vertaald naar M),
  - aggregeren van rijen (SUM(), COUNT(), AVG())
  - afgeleide waarden berekenen (bedrag = prijs * aantal)
  - filteren van dubbele waarden, NULL waarden, fouten, etc.
  - draaien van tabellen (velden voorkeur1 en voorkeur2 splitsen in 2 records)
- **Load**: bepalen hoe de data wordt ingevoerd in het datawarehouse.
  - bestaande waarden overschrijven
  - alleen nieuwe waarden toevoegen
  - nieuwere waarden verversen

controleerbaar herhaalbaar

Het ETL-proces moet controleerbaar en herhaalbaar zijn. Controleerbaar omdat corrupte gegevens funest zijn voor de kwaliteit van de analyse, vooral als niet tijdig wordt ontdekt dat de gegevens in het datawarehouse niet kloppen. Herhaalbaar omdat gegevens in de bronnen kunnen veranderen en worden uitgebreid door de tijd.

SQL-technieken

In Oracle kunnen we ETL-schema's maken in de vorm van mappings. Eigenlijk is een mapping een visueel schema van twee of meer SQL-queries. Één of meer voor Extract en Transform en één voor het Loaden in de datawarehouse tabel. Bij het maken van mappings kunnen we dus gebruiken maken van SQL-technieken om de data te bewerken. In de ETL-schema's modelleren we dan ook SQL-technieken om de gewenste bewerkingen uit te voeren.

7.2 ETL-schema's

correct ontwerp

ETL is geen trial-and-error aangelegenheid, want dat kan leiden tot een datawarehouse waar corrupte gegevens in staan, terwijl je dat soms zelf niet eens in de gaten hebt. Een zorgvuldig correct ontwerp van het ETL-proces is nodig om verzekerd te zijn van succes. De specificaties voor het ETL-proces maak je in de vorm van een ETL-schema (fig. 7.2.1). Een ETL-schema beschrijft de relatie tussen gebruiken brontabellen en één doeltabel, met alle noodzakelijke bewerkingen (inclusief het koppelen van brontabellen)

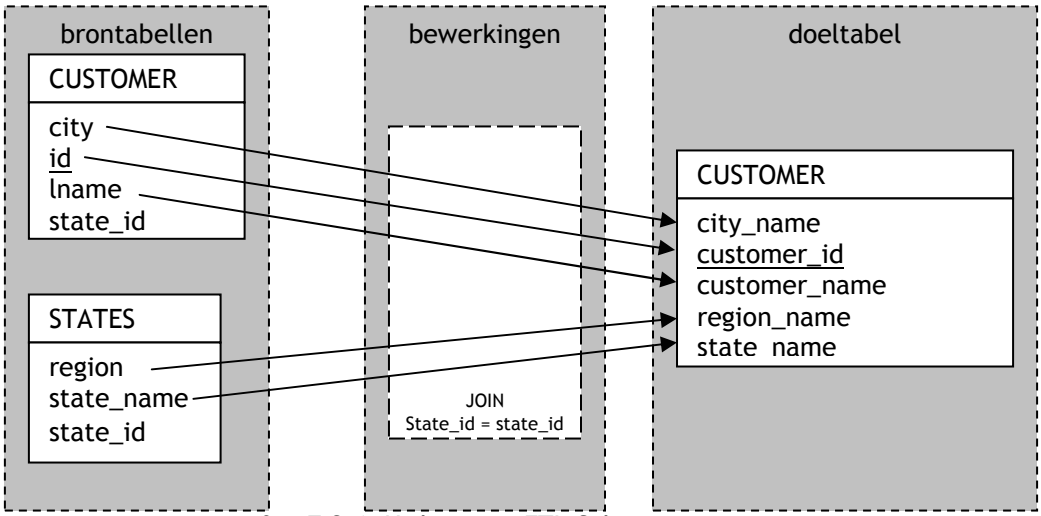


fig. 7.2.1 Uitleg over ETL-Schema

ETL schema
Brontabel
key-source-table
kwaliteit controle
join
doeltabel
pijl
Identificerende attributen

Richtlijnen:

- **Doeltabel**: Elk ETL-schema heeft één doeltabel (feitentabel of dimensie in het datawarehouse) en die wordt ten behoeve van de leesbaarheid altijd rechts gezet.
- **Brontabellen** staan altijd links. De **bovenste brontabel** wordt ook wel de **key-source-table** genoemd. De key-source-table bevat hetzelfde graan niveau als de doeltabel (elk record wordt verwerkt tot een record in de datawarehouse tabel).
- **Join-pad**: Bij elk record uit de key-source-table hoort maar één record uit elke van de andere tabellen. Vuistregel is dan ook dat vanuit de key-source-table een join-pad moet bestaan naar de **volledige primary key** van elke tabel die erbij wordt gejoind. (uitzondering kan voorkomen als er geaggregeerd wordt).
- **Pijlen**: Een pijl staat symbool voor ETL, die symboliseert het ophalen uit de brontabel, eventueel bewerken en opslaan in de doeltabel.
- **Bewerkingen bij een enkele pijl**. Bij een pijl mogen bewerkingen worden geschreven die erop worden uitgevoerd.
- **Bewerkingen over de gehele output**. Bewerkingen op meer dan één pijl zoals join, union, group by, worden weergegeven door een apart gestippeld vak over de pijlen te tekenen. Benodigde condities worden daarbij gezet.
- **Identificerende atributen**. De onderstreepte attributen in de doeltabel zijn de identificerende attributen (dat hoeft niet de primaire sleutel van de doeltabel te zijn!). Aan de hand van die identificerende attributen kan in een ETL-script worden bepaald of een record uit de bron al bestaat in het datawarehouse of als nieuw record moet worden toegevoegd. Over het algemeen zijn alleen de volledig overgenomen primary key attributen uit de key-source-table de identificerende attributen in de doeltabel. Uitzondering daarop is de kleinst mogelijke unieke combinatie van foreign keys bij een geaggregeerde feitentabel.

stap 6: ETL schema

In stap 7 maken we per tabel in het sterschema minstens één ETL-schema. Deze gezamenlijke schema's geven aan hoe de data uit de bronnen naar het datawarehouse kan worden gekopieerd. In onderstaand schema is de key-source-table verbonden met twee andere brontabellen. In de doeltabel is omzet een afgeleid gegeven dat wordt berekend. Bewerkingen zoals berekeningen kunnen bij de pijl worden geschreven.

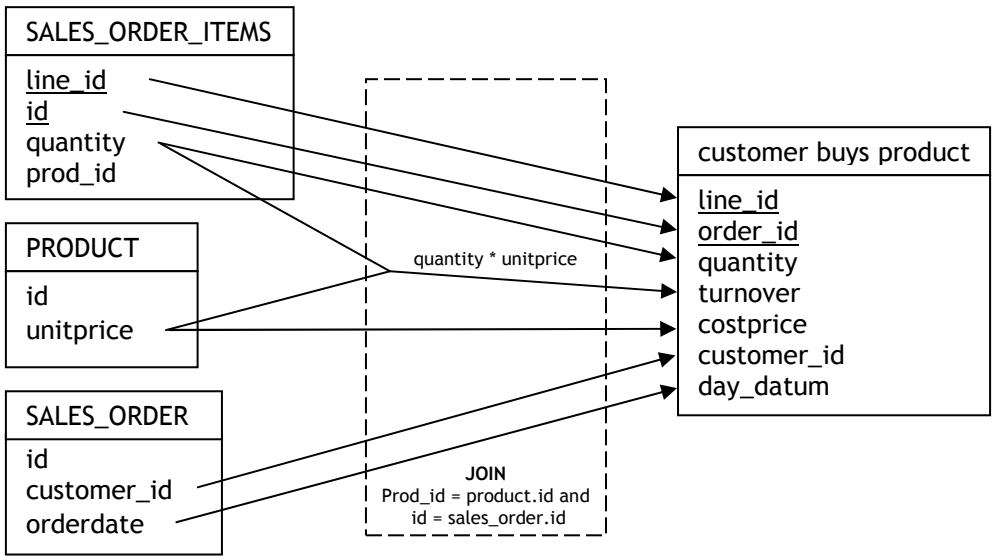


fig. 7.2.2 ETL-schema

conversiefunctie to_?()

Conversie van gegevens is een veel voorkomende bewerking. In fig. 7.3.1 staat bijvoorbeeld een bewerking van bonus_date naar een jaartal. Alle simpele bewerkingen die op één gegeven betrekking hebben zet je als functie bij de desbetreffende pijl. Conversies geef je altijd een naam met to_?(). Voor bewerkingen die meer rijen betreffen teken je een gestippelde rechthoek.

kwaliteit controle

Een goede kwaliteitscontrole is om na te gaan dat bij elk record uit de key-source-table maar één record hoort uit elk van de andere tabellen. Daarvoor mogen er alleen tabellen in de join worden toegevoegd door te verwijzen naar de volledige primaire sleutel (in dit geval product.id en sales_order.id). Dit geldt overigens niet als er een aggregatiefunctie wordt gebruikt.

7.3 Aggregatie

aggregate maken

Om een aggregate feitentabel kunnen maken moeten onderliggende feiten worden geaggregeerd (samengenomen). Dat kunnen we doen door groepen te formeren en per groep een geaggregeerde waarde te laten bereken. In SQL termen werken we bij een aggregatie dus met een group by in combinatie met aggregatiefuncties zoals SUM(), COUNT() en AVG(). In fig. 7.3.1 wordt gegroepeerd op year en emp_id (elke combinatie komt dus maar één keer voor) om vervolgens bonus_amount te kunnen sommeren.

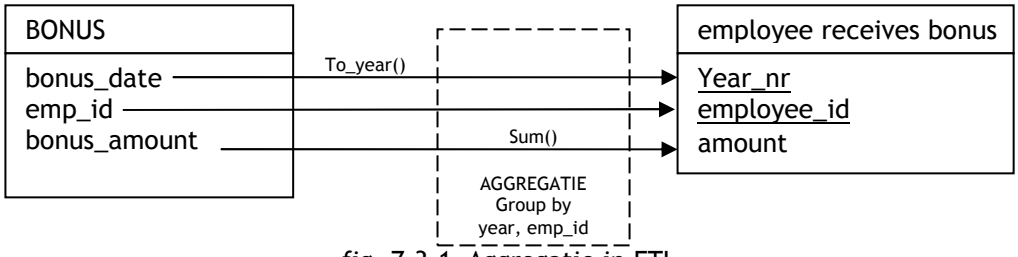


fig. 7.3.1 Aggregatie in ETL

7.4 Dimensies

primary key dimensie moet uniek zijn

In dimensietabellen mag elke primary key maar één keer voorkomen (zoals in elke databasetabel). Dat betekent dat we soms dubbele waarden eruit moeten filteren met distinct zoals in fig. 7.4.1 . Indien een dimensie niet in een aparte bron-tabel staat dan kunnen we via de key-source-table achterhalen welke dimensiewaarden echt gebruikt worden.

volgorde van bewerkingen maakt uit

ETL-schema's worden met de richting van de pijlen mee uitgevoerd, de volgorde van bewerkingen maakt dus uit. Bij figuur fig. 7.4.1 moet eerst de datum worden vertaald naar jaar en daarna pas een distinct worden gebruikt (omdat jaren niet dubbel voor mogen komen. Bij fig. 7.4.2 levert elke datum een dimensierecord op en kan uit een datum alle dimensiegegevens worden afgeleid.

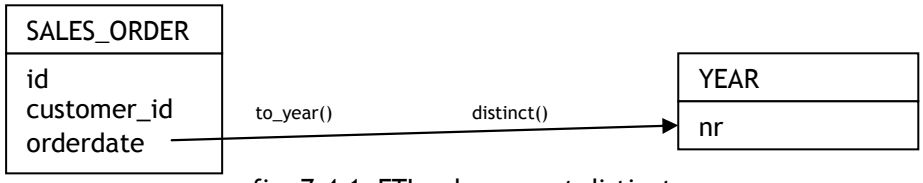


fig. 7.4.1 ETL-schema met distinct

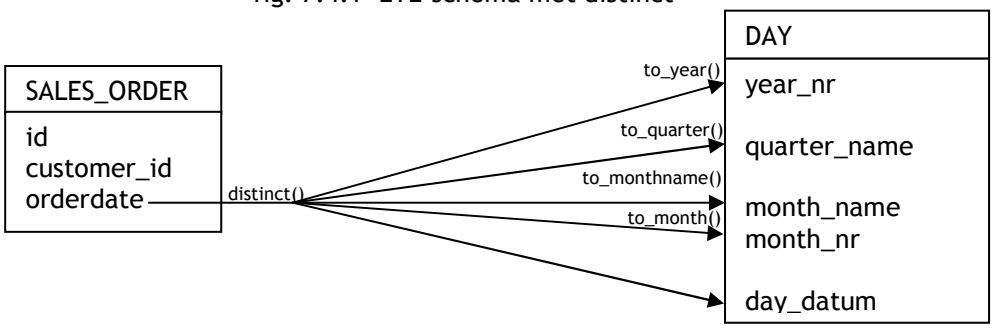


fig. 7.4.2 ETL-schema met eerst distinct

union

Een Conformed Dimension wordt door meer dan één feitentabel gebruikt. Dan kan het zijn dat de dimensierecords over verschillende brontabellen staan verdeeld. We kunnen een UNION gebruiken om alle voorkomende dimensiewaarden uit verschillende bronnen samen te voegen. NB. een UNION voert automatisch een distinct uit over de uitvoer.

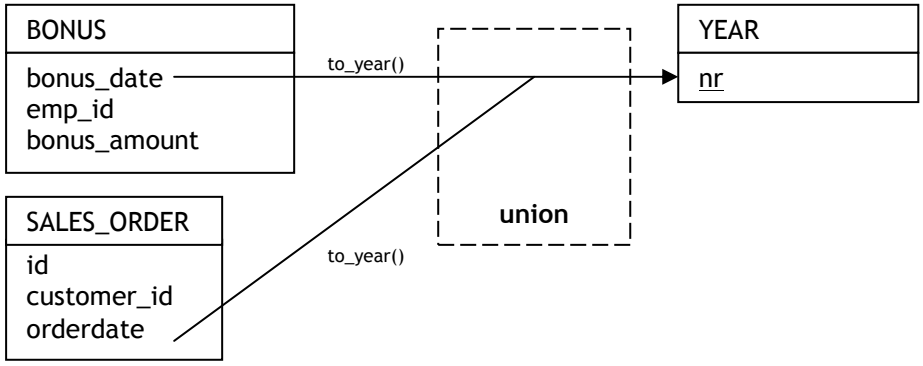


fig. 7.4.3 ETL-schema met UNION

doelattribuut moet een pijl hebben

Elk attribuut in de doeltabel moet een ingaande pijl hebben. Het diagram wordt van links naar rechts gelezen, dus eerst wordt de join uitgevoerd en dan pas de distinct.

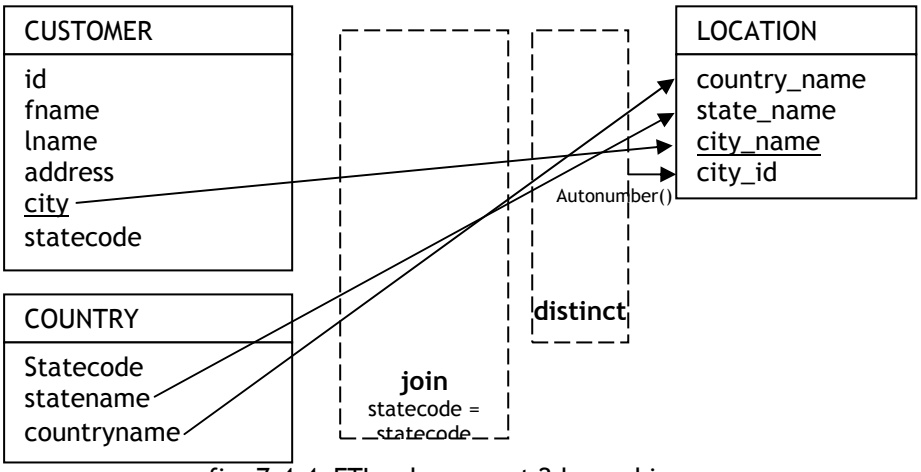


fig. 7.4.4 ETL-schema met 2 bewerkingen

autonumber

In de dimensie 'location' is de naam van de stad geen integer en dus minder geschikt als Primary Key. Daarom is city_id als kunstmatige (surrogaat-)sleutel toegevoegd. Vaak kiest men er voor om met een autonumber() functie van de database daar automatisch een uniek nummer aan toe te kennen.

surrogaatsleutels

Ter informatie: in theorie is het verstandig om elke dimensie te voorzien van een surrogaatsleutel. Daar zijn twee redenen voor:

- Primaire sleutels kunnen in de werkelijkheid nog wel eens veranderen. Denk aan een auto die een nieuwe kentekenplaat krijgt.
- slowly changing dimensions
  Attributen van een dimensie kunnen veranderen. We spreken dan van een Slowly Changing Dimension. Bv een auto is wit in 2004 en rood (overgespoten) in 2005 maar met hetzelfde kenteken.

We kunnen dit soort problemen oplossen door het gebruik van surrogaatsleutels, maar dat valt verder buiten de stof van deze cursus.

7.5 Testen

Correct ontwerp van ETL is cruciaal. Fouten in ETL-schema's leiden over het algemeen wel tot het vullen van tabellen, maar de data die erin komt is corrupt.

Door het gebruik van vaste richtlijnen bij het ETL ontwerp (zie 7.2), worden een hoop fouten voorkomen. De leesbaarheid wordt ermee verhoogd en sommige fouten worden vermeden. Daarnaast kan men het ETL-proces testen, en achteraf controleren of het resultaat conform verwachting is.

Een testset bestaat zoals altijd uit een beschrijving van de beginsituatie (concrete records in brontabellen én doeltabellen) en het daarbij behorende gewenste eindresultaat (weer in termen van concrete records). In praktijk doet men er goed aan om het ETL-proces na elke aanpassing eerst te testen voordat die gebruikt wordt. Uitvoeren van een incorrect ETL-proces kan immers leiden tot corruptie van het hele datawarehouse.

Vóór het uitvoeren van ETL-schema's moet men van tevoren uitzoeken hoeveel rijen er geladen en geüpdate moeten worden. Aan de ene kant kan men dat gebruiken om te bepalen hoeveel tijd het ETL-proces in beslag zal nemen. Aan de andere kant kan men aan da hand daarvan achteraf controleren of het ETL-proces correct is verlopen.

8 Datamart

8.1 Analyse problemen

grootte datawarehouses

Datawarehouses kunnen heel groot worden. Een datawarehouse omvat grofweg de hele dataset waar je Business Intelligence op toe wilt kunnen passen. Bij organisaties met een hoog ambitie en maturity niveau ten aanzien van B.I. kan dat zo'n beetje alle digitale data omvatten waar intelligence uit te putten is. In zo'n datawarehouse zit naast de huidige operationele data ook behoorlijk wat historie opgeslagen om onderzoek over langere tijd mogelijk te maken. Ahold heeft waarschijnlijk het grootste commerciële datawarehouse in Nederland met naar schatting 12TB (een TeraByte = 1000GB). Wall-Mart loopt wereldwijd voorop en is momenteel (juni 2006) haar datawarehouse capaciteit aan het upgraden naar 900TB. Het huidige datawarehouse bedraagt echter 'slechts' 550TB.

performance problemen

Probleem met zulke grote datawarehouses is dat het beantwoorden van queries teveel tijd kost. Dat wordt ondervangen door voor specifieke doelen aparte BI-datamarts te maken. Elke datamart bevat alleen de voor dat doel benodigde data. Dus minder historie en minder detail. De datamarts worden gevuld vanuit het datawarehouse zoals in onderstaande figuur te zien is:

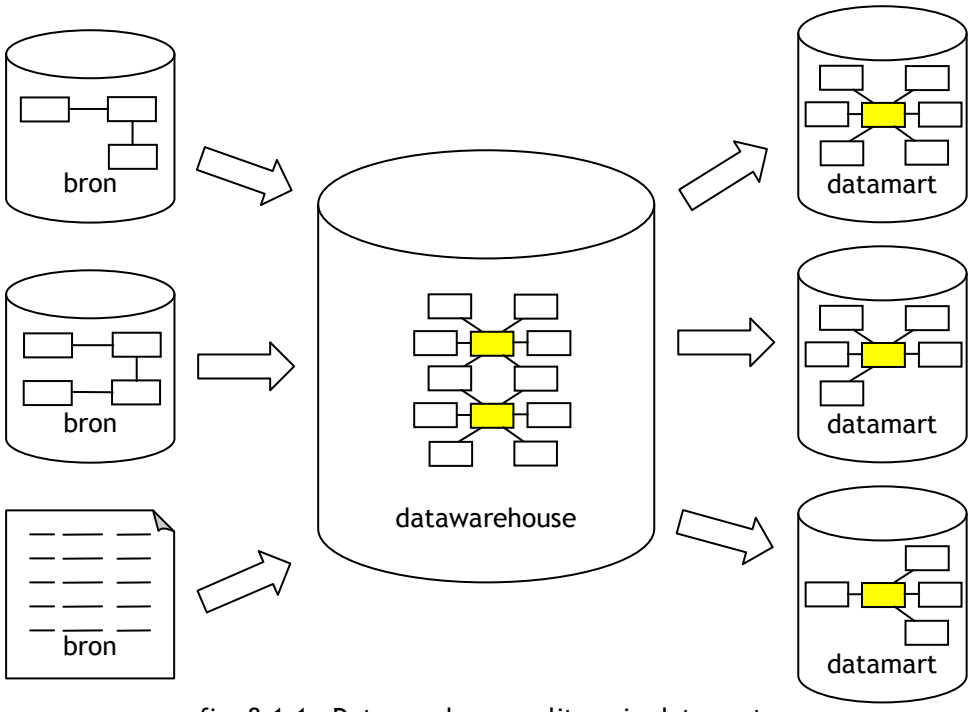


fig. 8.1.1 Datawarehouse splitsen in datamarts

8.2 Datawarehouse

modelleren voor opslag

Het datawarehouse wordt ontworpen via een (set) sterschema('s). Indien men intelligence niet rechtstreeks uit het datawarehouse gaat halen, maar uit datamarts, dan kan het datawarehouse het best gemodelleerd is voor opslag:

- probeer zoveel mogelijk details te behouden (Atomic feiten)
- neem verschillende typen gebeurtenissen op in verschillende feitentabellen
- koppel alleen dimensies die een directe relatie hebben met het feit
- los alle informatiekwaliteit problemen op

De voordelen van dit omvangrijke en minder complexe datawarehouse zijn dat het minder druk legt op operationele bronnen tijdens het ETL-proces en dat het datawarehouse een zo correct en volledig mogelijk archief wordt.

minder analyse mogelijkheden

De nadelen van een omvangrijk en minder complex datawarehouse zijn dat directe analyse in mindere mate mogelijk is. Vooral voor businessvragen die betrekking hebben op een combinatie van gebeurtenissen of die betrekking hebben op niet direct verbonden dimensies moet men eerst een datamart afleiden. Niet elke vraag is daardoor ad-hoc te beantwoorden; indien er nog geen geschikte datamart is, zal er een moeten worden gecreëerd.

alternatieven voor sterschema

In de praktijk zien we steeds vaker dat de noodzaak om een datawarehouse als sterschema te implementeren afneemt. Als analyse daar toch niet rechtstreeks op wordt gedaan kan men ook kiezen voor een makkelijkere relationele implementatie of bv. een 'data-vault' (een techniek waar we verder niet op ingaan). In de toekomst kunnen tools wellicht wel omgaan met een combinatie van feitentabellen en indirecte dimensies, waardoor analyse direct op het datawarehouse mogelijk wordt. Dat pleit enigszins voor het gebruik van een sterschema bij het modelleren van datawarehouses.

8.3 Datamarts

datamart

Een Datamart is een stukje datawarehouse dat is ingericht voor een specifiek doel. De term is nogal verwarrend omdat die ook wordt gebruikt om zgn. staging area's aan te duiden die aan de voorkant worden gebruikt om Datawarehouses te vullen. Wij hebben het echter alleen over een Datamart als afgeleide uit een datawarehouse waarop men met BI-tools kan werken. Het neemt vaak de plaats in van periodieke rapportage en stelt de eindgebruiker in staat zelf overzichten te maken met een query-, report- of OLAP-tool.

omvang terugbrengen

Afhankelijk van het doel bevat een datamart ten opzichte van het datawarehouse doorgaans:

- minder tabellen
- minder historische data
- geaggregeerde data

beperkingen van BI-tools

Een datamart kan ook een middel zijn voor de BI-analist die bepaalde verbanden wil onderzoeken met OLAP of data-mining tools. Ons 'perfecte' datawarehouse bevat wel alle data en relaties, maar de meeste BI-tools kunnen daar maar beperkt analyse op doen. Er zijn beperkingen met betrekking tot:

- het combineren van verschillende gebeurtenissen (die in verschillende feitentabellen staan)
- het gebruik van dimensies die niet direct aan een feitentabel hangen
- NULL–waarden bij foreign keys naar dimensies

aanpassingen ten opzichte van datawarehouse

Een datamart wordt net als een datawarehouse in een sterschema beschreven. De mogelijkheden voor analyse kunnen worden vergroot door het ontwerp daarop aan te passen. Veel voorkomende aanpassingen bij het ontwerpen van een datamart zijn:

- Aggregate maken op een hoger graan niveau.
- Verschillende feiten combineren tot één feitentabel (aangezien de meeste BI-tools redelijkerwijs maar één feitentabel tegelijk kunnen onderzoeken).
- Indirect verbonden dimensies direct koppelen aan de te onderzoeken feitentabel.

stap 7: datamart

Per datamart moet een apart sterschema en ETL-schema worden gemaakt.

8.4 Herontwerp feitentabel

datamart sterschema

In veel gevallen zijn datamarts eenvoudige ontwerpen die te maken zijn aan de hand van de regels voor datawarehouse ontwerp. Sommige businessvragen zijn echter niet te beantwoorden op eenvoudige datamarts. Soms moeten feitentabellen worden gecombineerd, verlaagd tot dimensie of worden verbonden aan indirecte dimensies (via een ander feit). We gaan nu een aantal technieken bespreken die door een aangepast ontwerp en ETL het mogelijk maken om de tekortkomingen van OLAP-tools, om bepaalde vragen te kunnen onderzoeken, te omzeilen.

multi-cube analyse

De meeste OLAP-tools geen zogeheten multi-cube analyse, en kunnen dus geen vragen over meer dan één feitentabel tonen. Enkele pakketten doen dat wel maar beperkt, waardoor alleen heel basale vragen over meer dan één feitentabel kunnen worden onderzocht. Indien we bonussen in combinatie met verkoopfeiten willen onderzoeken (bv of verkopers na het krijgen van bonussen meer gaan verkopen) dan moeten we twee feitentabellen uit het datawarehouse combineren tot één feitentabel in een datamart.

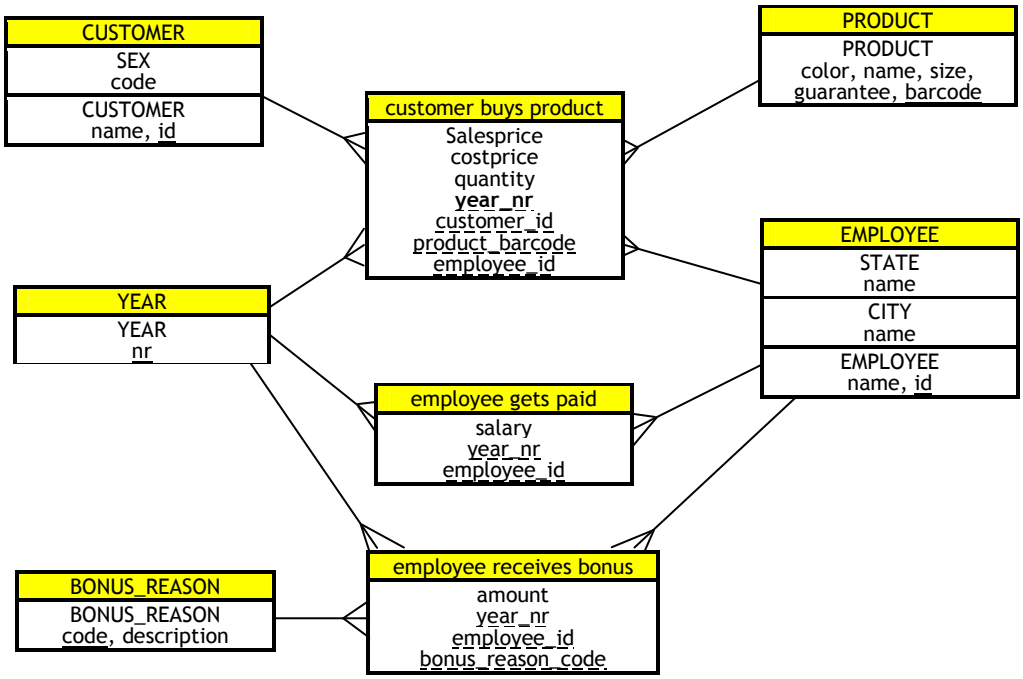


fig. 8.4.1 Datawarehouse met 3 feitentabellen

feiten verenigen via conformed dimensions

Bij het verenigen van meetwaarden uit twee feitentabellen gelden een aantal vuistregels. Je kunt twee feitentabellen alleen samennemen door conformed dimensions te gebruiken. De vuistregels zijn afhankelijk van de multipliciteit op recordniveau.

8.5 Verenigen van 1-op-0..1 verbonden feitentabellen

1-op-0..1 feitentabellen zijn prima samen te nemen. Zoals bij SALARIS en BONUS. De meetwaarden salaris en bonus kunnen zonder probleem worden overgenomen.

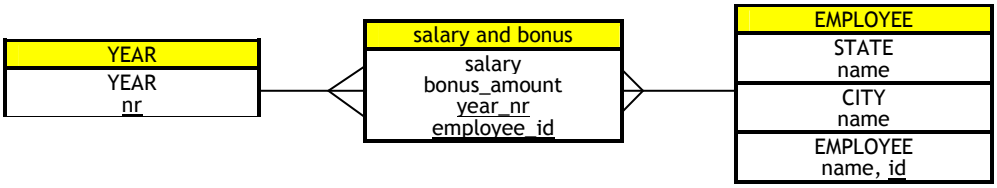


fig. 8.5.1 Vereniging van 1-op-1 verbonden feitentabellen

Het echte werk gebeurt met ETL. De kosten kunnen laag blijven door de feitentabellen van het datawarehouse te gebruiken als bron.

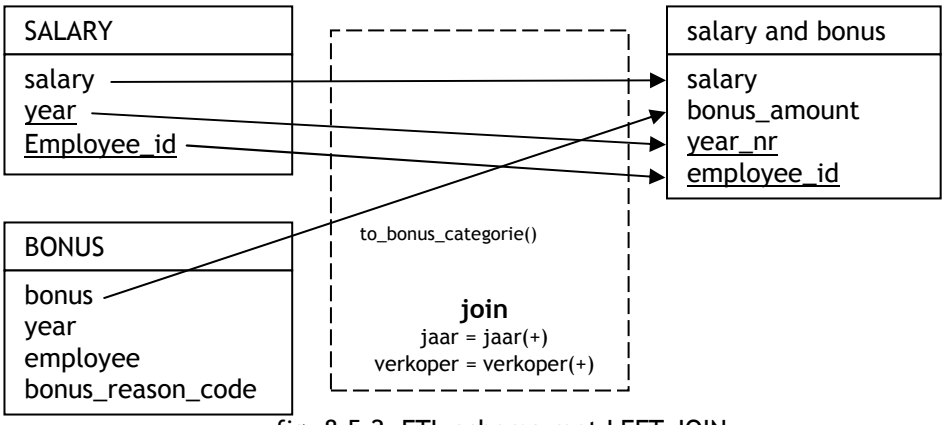


fig. 8.5.2 ETL-schema met LEFT JOIN

Bij een 'normale' JOIN verdwijnen salarissen waarbij geen bonus hoort. Omdat niet elke verkoper elk jaar een bonus krijgt, is een zgn. LEFT JOIN nodig. Die zorgt ervoor dat alle records uit de eerste tabel (SALARIS) in het resultaat voorkomen, ongeacht of er een record uit de tweede tabel (BONUS) bij hoort. De LEFT JOIN wordt in Oracle gelegd door (+) achter het eventueel ontbrekende attribuut te zetten. Bij andere DBMS-en gebeurt dat op een andere manier.

8.6 Verenigen van 1-op-N verbonden feitentabellen

1-op-N feitentabellen zijn beperkt samen te nemen. De tabellen BONUS en VERKOOP zijn 1-op-N verbonden omdat er bij één jaar en één verkoper maar één BONUS hoort, maar N verkopen. De meetwaarden uit de 1-tabel (hier BONUS) kunnen niet worden toegevoegd aan de feiten in de N-tabel. Dan zou dezelfde bonus immers meerdere keren in de feitentabel voorkomen waardoor de totale bonus niet meer klopt. Men kan in plaats daarvan een dimensie opnemen die aangeeft of een verkoopfeit gerelateerd is aan een bonus. In die dimensie kan men wel categorieën opnemen die een indicatie van de hoogte van de bonus geven, maar het is geen feit meer waarmee gerekend mag worden. Intelligence kan toch verkregen worden omdat de resultaten bij het krijgen van bonussen kunnen worden vergeleken met de resultaten zonder bonus.

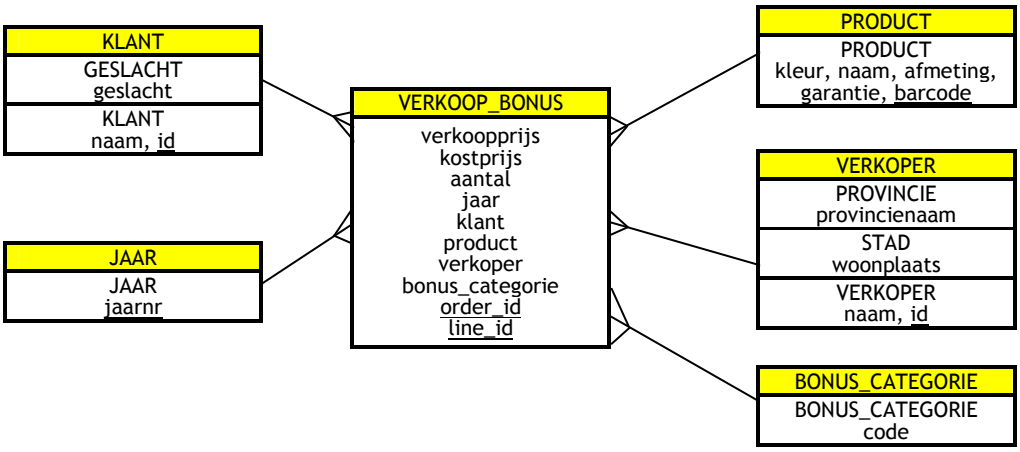


fig. 8.6.1 Sterschema met verlaagde dimensie

Het ETL-schema zal nu uit twee delen bestaan. Één voor het vullen van de nieuwe dimensie en één voor het vullen van de afgeleide feitentabel.

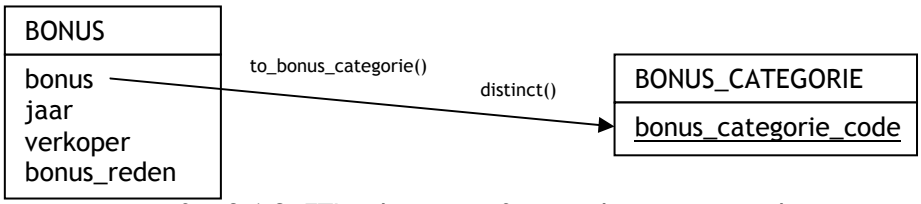


fig. 8.6.2 ETL-schema om feit tot dimensie te verlagen

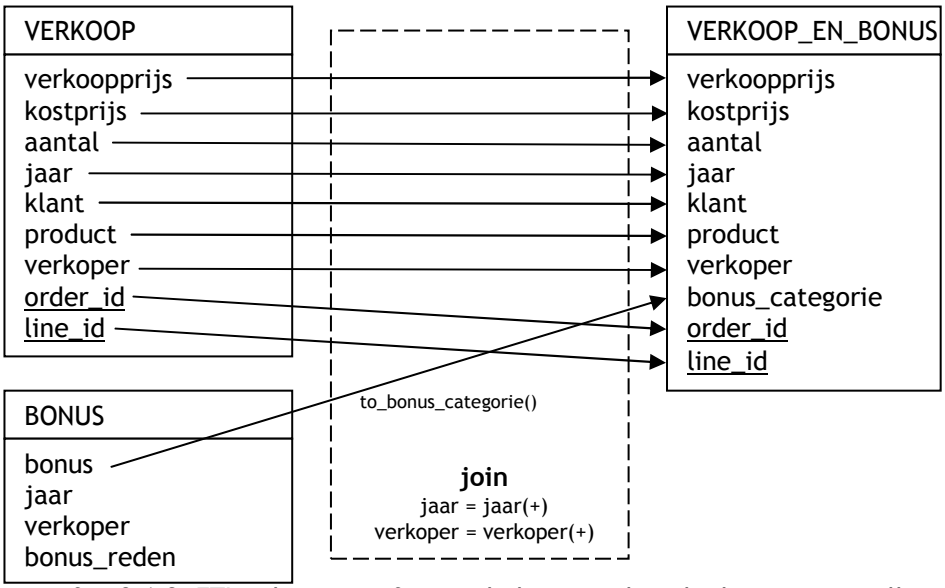


fig. 8.6.3 ETL-schema om feitentabel met verlaagde dimensie te vullen

expressie om meetwaarde om te zetten in dimensiewaarde

In beide ETL-schema's komt een expressie voor die het bonusbedrag omzet in een bonus_categorie. Wat die expressie is, is niet belangrijk, als in beide ETL-schema's maar dezelfde expressie staan. Een alternatief is bij BONUS een boven en ondergrens op te nemen, en bij het vullen van de feitentabel de bonus te vergelijken met de boven en ondergrenzen in de dimensietabel.

aggregeren tot 1-op-1

Er is een alternatief zijn voor het verenigen van 1-op-N verbonden feitentabellen. Indien in het in paragraaf 8.5 gegeven voorbeeld bij een salaris meer dan één bonus kan horen, kunnen we ervoor kiezen om de bonussen eerst samen te nemen zodat er wel een 1-op-0..1 situatie ontstaat. Dat kan in één ETL-schema door eerst een aggregatie te gebruiken om de bonussen per jaar en verkoper samen te nemen en het resultaat daarvan in de LEFT JOIN te stoppen.

Verlies non-conformed dimensies

Die truc kunnen we niet zondermeer gebruiken voor VERKOOP_BONUS, want dan raken we de dimensies KLANT en PRODUCT kwijt. De oplossing die hiervoor is beschreven verlaagt als het ware de bonusfeiten tot een dimensiecategorie, zodat onderzoek naar de invloed van bonussen enigszins mogelijk wordt in combinatie met de KLANT en PRODUCT dimensies.

8.7 Verenigen van N-op-N verbonden feitentabellen

N-op-N verbonden feitentabellen zijn zeer beperkt samen te nemen. Dat zou in het voorbeeld van paragraaf 8.6 het geval zijn als een verkoper verschillende bonussen per jaar kan ontvangen. Dezelfde twee constructies zijn mogelijk.

- De feiten uit één of beide feitentabellen aggregeren zodat een 1-op-N of 1-op-1 verband ontstaat. De consequentie is wel dat alle non-conformed dimensies (bonus_reden, klant en product) komen te vervallen.
- De meetwaarden uit een van de twee feitentabellen aggregeren en 'verlagen' tot dimensie van de andere feitentabel.

8.8 Toevoegen van afgeleide dimensies

afleiden van dimensies

In een datawarehouse neem je in eerste instantie alleen dimensies op die direct met feiten verbonden zijn. Een klant koopt een product in een filiaal bij en verkoper op een bepaald tijdstip. KLANT, PRODUCT, FILIAAL, VERKOPER en TIJD zijn dimensies die rechtstreeks met het feit verbonden zijn. Via de feitentabel BONUS kan echter ook de reden worden achterhaald waarom een bonus is ontvangen. Bij een orderregel hoort immers één jaar, één verkoper, maximaal één bonus en dus maximaal één bonus_reden.

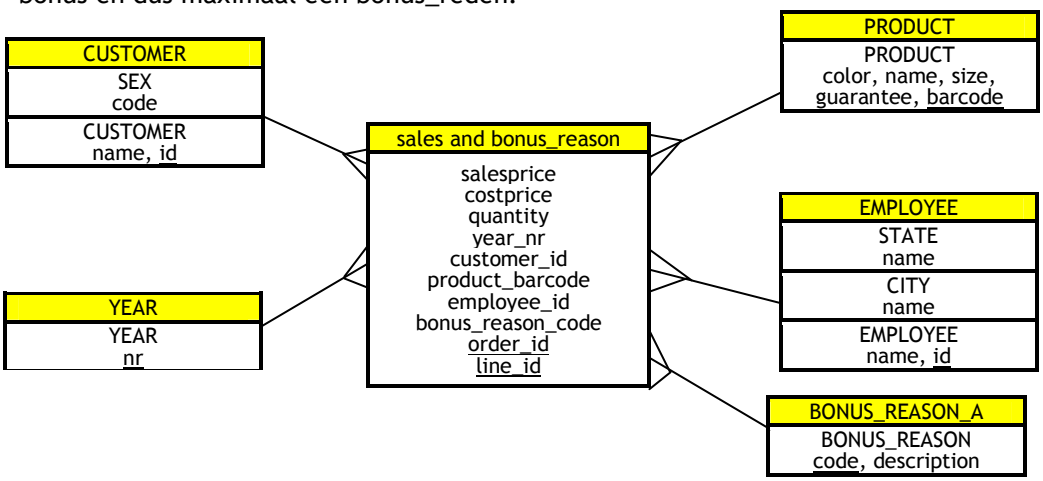


fig. 8.8.1 Datamart met afgeleide dimensie

NULL-waarden als Foreign Key

Er zit echter één adder onder het gras. De Foreign Key naar BONUS_REDEN kan NULL-waarden bevatten. Dat kan problemen opleveren met het gebruik, omdat een INNER JOIN met BONUS_REDEN ervoor zorgt dat sommige feiten verdwijnen. De verkooptotalen kloppen dan niet meer. In de regel doet men er goed aan om NULL-waarden bij Foreign Keys zoveel mogelijk te vermijden.

NULL-waarden vervangen door no-NULL waarde

Om dat te verhelpen kun je ervoor kiezen om een extra waarde aan de BONUS_REDEN dimensie toe te voegen die bijvoorbeeld 'geen bonus' heet. Zodoende kunnen alle verkoopfeiten aan de BONUS_REDEN dimensie worden gekoppeld.

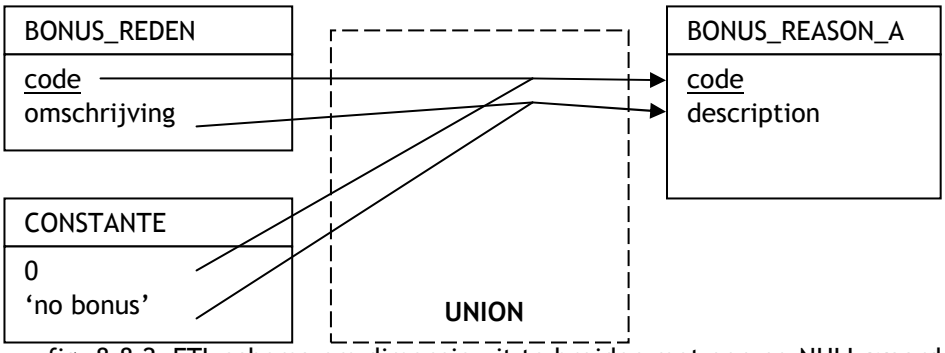


fig. 8.8.2 ETL schema om dimensie uit te breiden met een no-NULL waarde

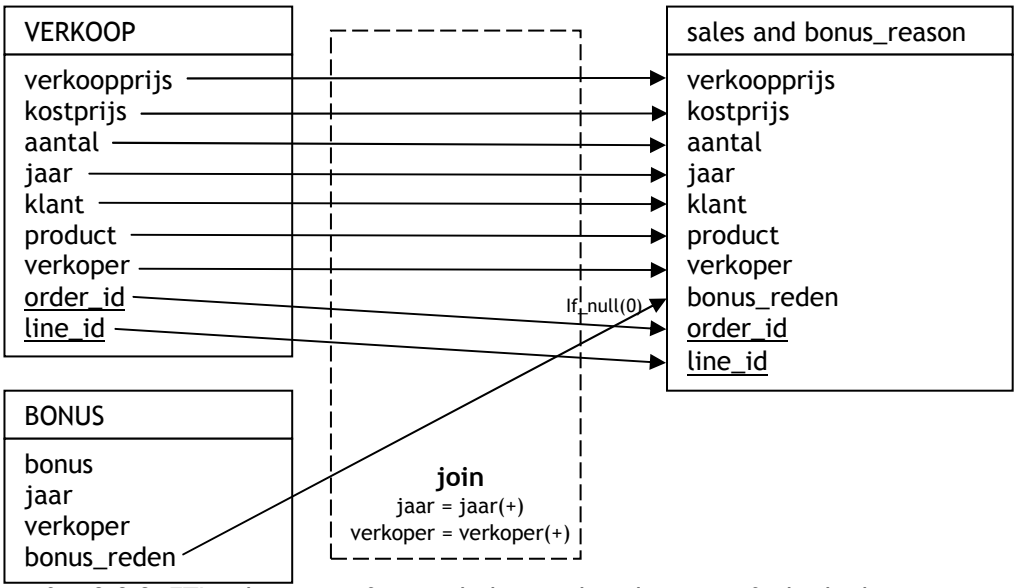


fig. 8.8.3 ETL-schema om feitentabel uit te breiden met afgeleide dimensie

Oracle staat op SQL-niveau niet toe dat meer dan 2 tabellen worden verbonden met een LEFT JOIN. Indien het nodig is om meer afgeleide dimensies aan een feitentabel toe te voegen moet er per afgeleide dimensie een apart ETL-schema's worden opgesteld.