HO
GENT

Faculteit Bedrijf en Organisatie

Een vergelijkende studie tussen Data Vault 2.0 en het dimensioneel model bij het modelleren van een data warehouse.

Jorik Spiesschaert

Scriptie voorgedragen tot het bekomen van de graad van
professionele bachelor in de toegepaste informatica

Promotor:
Stijn Lievens
Co-promotor:
Jochen Stroobants

Instelling: DHL Pharma Logistics

Academiejaar: 2018-2019

Tweede examenperiode

Faculteit Bedrijf en Organisatie

Een vergelijkende studie tussen Data Vault 2.0 en het dimensioneel model bij het modelleren van een data warehouse.

Jorik Spiesschaert

Scriptie voorgedragen tot het bekomen van de graad van
professionele bachelor in de toegepaste informatica

Promotor:
Stijn Lievens
Co-promotor:
Jochen Stroobants

Instelling: DHL Pharma Logistics

Academiejaar: 2018-2019

Tweede examenperiode

Woord vooraf

Voor het bekomen van een professionele bachelor in de Toegepaste Informatica dient een bachelorproef geschreven te worden.

Het onderwerp van deze bachelorproef werd mij aangebracht door Jochen Stroobants bij de start van mijn stage. Als stageopdracht diende een data warehouse opgesteld te worden aan de hand van Data Vault 2.0. Zelf was ik kritisch tegenover deze keuze en ik vroeg mij dan ook af of Data Vault 2.0 de juiste keuze was ten opzichte van het dimensioneel model. Het leek mij zeer interessant om de vergelijking te maken tussen beide modellen en om te concluderen of het dimensioneel geen betere keuze was.

Graag bedank ik in eerste instantie mijn promotor Stijn Lievens voor de uitgebreide feedback die ik ontving tijdens het opstellen van een voorstel en tijdens het schrijven van deze paper. Door zijn kritische kijk kon ik de kwaliteit van deze paper naar een hoger niveau brengen.

Anderzijds bedank ik graag het bedrijf Cubis Solutions, omdat ik gebruik kon en mocht maken van hun infrastructuur. In het bijzondere wil ik graag Jochen Stroobants, Sven Van Rillaer en Sander Allert bedanken voor hun kennis en feedback die ik ontving tijdens het opstellen van deze paper.

Samenvatting

DHL Pharma Logistics wil een data warehouse aanmaken om hun rapporteringen te automatiseren. De data warehouse wordt gemodelleerd via de Data Vault 2.0 methodologie. In deze paper wordt onderzocht of modelleren via deze manier wel degelijk de juiste keuze was ten opzichte van het dimensioneel model.

Deze paper kan nuttig zijn voor informatici die actief zijn of interesse hebben in Business Intelligence. Er is geen specifieke voorkennis rond Data Vault of het dimensioneel modelleren nodig om door deze paper te gaan. Dit vergelijkend onderzoek kan informatici helpen bij het beslissen van een data model methodologie.

Vooraleer het onderzoek van start gaat, wordt een literatuurstudie weergegeven met de informatie en begrippen die nodig zijn om het volledige onderzoek mee te kunnen volgen.

Vervolgens wordt in het onderzoek twee data warehouses opgebouwd op basis van dezelfde data. Er zal een data warehouse gemodelleerd worden via Data Vault 2.0, de andere data warehouse zal gebaseerd zijn op het dimensioneel model. Deze data warehouses zullen van later in dit onderzoek gebruikt worden om beide data modellen te vergelijken op basis van performantie en schaalbaarheid.

Het vergelijkend onderzoek wordt uitgevoerd op basis van vijf pijlers: performantie, complexiteit, flexibiliteit, schaalbaarheid en audit. Zowel Data Vault 2.0 als het dimensioneel model worden onderworpen aan deze vijf pijlers, op basis van de noden van DHL Pharma Logistics wordt een juiste conclusie opgemaakt.

Als resultaat in dit onderzoek blijkt dat het kiezen voor het opstellen van een data warehouse aan de hand van dimensioneel modelleren een betere keuze was geweest.

Naar de toekomst toe kan er onderzocht worden of Data Vault 2.0 een juiste keuze kan zijn bij het opstellen van big data modellen.

Inhoudsopgave

Lijst van figuren

Lijst van tabellen

1. Inleiding

Beslissingen genomen door het management worden vaak gemaakt op basis en ondersteund door van data en rapporteringen. Bij sommige bedrijven worden de benodigde data nog steeds manueel uitgerekend. Deze methodologie heeft enkele nadelen: kans op fouten, tijdrovend, ... Bedrijven die dit proces willen automatiseren en digitaliseren moeten hiervoor een data warehouse opbouwen. Data afkomstig uit verschillende bronnen worden dan ingeladen in één centrale plaats. Op basis van de data afkomstig uit een data warehouse kan een rapporteringsomgeving de data visualiseren, dashboards en rapporteringen opmaken. Het is gebruikelijk dat het datamodel in een data warehouse opgebouwd wordt aan de hand van een dimensioneel model. Dit kan ook gebeuren aan de hand van een ander model, genaamd Data Vault.

Bij DHL Pharma Logistics wordt een data warehouse opgebouwd aan de hand van het Data Vault model. Maar waar zitten de verschillen bij Data Vault in vergelijking met het dimensioneel model? Was de keuze voor Data Vault bij het modelleren van de data warehouse juist voor DHL Pharma Logistics?

1.1 Probleemstelling

Wie interesse heeft in Business Intelligence of modelleren van data, is ongetwijfeld al in aanraking gekomen met het dimensioneel model. Maar dit is heus niet de enige techniek die beschikbaar is voor het modelleren van datamodellen. Data Vault 2.0 werd al geïmplementeerd bij bedrijven zoals IBM, Oracle en ING. Wat zouden de redenen kunnen zijn waarom deze bedrijven kiezen voor Data Vault? Met welke elementen onderscheidt Data Vault zich ten opzichte van het dimensioneel model? Dit onderzoek is gericht naar

BI consultants en alle belanghebbenden.

1.2 Onderzoeksvraag

In dit onderzoek wordt een vergelijkende studie uitgevoerd tussen het dimensioneel modelleren en Data Vault 2.0, toegepast op een case bij DHL Pharma Logistics. Er wordt onderzocht waar de verschillen zitten bij het modelleren met Data Vault 2.0 en het dimensioneel modelleren. Er zal een antwoord trachten gevonden te worden op volgende deelvragen:

- **Performantie:** is er een significant verschil tussen de performantie tussen beide modellen?
- **Audit:** hoe wordt er omgegaan met de traceerbaarheid in beide modellen?
- **Schaalbaarheid:** hoe wordt er omgegaan met grote volumes data?
- **Complexiteit:** zijn beide modellen makkelijk begrijpbaar (voor IT en business)?
- **Flexibiliteit:** hoe makkelijk kunnen wijzigingen/toevoegingen gemaakt worden in beide systemen?

1.3 Onderzoeksdoelstelling

DHL Pharma Logistics wil een data warehouse ontwerpen om hun rapporteringen te automatiseren. De data warehouse zal gemodelleerd worden via Data Vault 2.0. In dit onderzoek zal bewezen worden als het dimensioneel model hier een betere keuze was geweest.

1.4 Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:

In hoofdstuk 2 wordt een overzicht gegeven van de stand van zaken binnen het onderzoeksdomein, op basis van een literatuurstudie.

In hoofdstuk 3 wordt beschreven wat de benodigdheden zijn voor dit onderzoek en worden twee data warehouses stap voor stap opgebouwd. Er zal een data warehouse gebaseerd op het dimensioneel model, de andere data warehouse gebaseerd op Data Vault. Het ETL-proces (2.1.6) wordt neergepend in deze paper voor zowel Data Vault als voor het dimensioneel model.

In hoofdstuk 4 zal een vergelijkend onderzoek gevoerd worden tussen Data Vault en het dimensioneel model. De onderzoeksvraag en deelvragen worden in dit hoofdstuk beantwoord.

In hoofdstuk 5, tenslotte, wordt de conclusie gegeven en een antwoord geformuleerd op de onderzoeksvragen. Daarbij wordt ook een aanzet gegeven voor toekomstig onderzoek binnen dit domein.

2. Stand van zaken

Dit hoofdstuk bevat een literatuurstudie omtrent data warehousing. Na het lezen van dit hoofdstuk zullen begrippen zoals dimensioneel modelleren, Data Vault 2.0 en data warehousing jou niet meer onbekend zijn en zal duidelijk zijn waarom er nood is aan data warehousing. Je zal in staat zijn om een eerste Data Vault model en een dimensioneel model te schetsen.

2.1 Inleiding data warehousing

> Torture the data, and it will confess to anything.
>
> *Ronald Coase*
> *Winnaar Nobelprijs in Economie (1991)*

Veel moderne, digitale bedrijven genereren tegenwoordig enorme volumes data. Deze data kunnen afkomstig zijn uit verschillende bronnen: ERP/CRM-systeem, flat-files (bijvoorbeeld Excel-sheets), big data, ... Bestuursleden gebruiken data om beslissingen te nemen die de onderneming toelaat om te (blijven) groeien of om bepaalde problemen op te sporen. Stel dat een onderneming meer kosten heeft dan opbrengsten, dan kunnen we op basis van alle gegevens die het bedrijf bezit een analyse maken. Zijn er overbodige kosten? Worden onze producten/diensten aan een te lage prijs verkocht? Dit zijn maar enkele vragen die kunnen opgelost worden wanneer het bestuur de correcte rapporteringen ontvangt. Ook wordt een data warehouse gebruikt om gegevens aan te leveren om budgetten op te stellen voor de komende jaren en om van forecasting gebruik te maken. Zo kan er bijvoorbeeld een

beslissing gemaakt worden welk budget er moet toegekend worden aan de verschillende afdelingen.

Om alles nog even samen te vatten: een data warehouse levert data aan een rapporteringsomgeving. Op basis van deze data worden beslissingen genomen.

2.1.1 Soorten data

Er kan een onderscheid gemaakt worden tussen verschillende soorten data. Voornamelijk kunnen we informatie opdelen in drie categorieën: gestructureerde, semi-gestructureerde en ongestructureerde data (Buneman, 1997). Volgens Langseth, Vivatrat en Sohn (2005), bestaat 95% van de globale informatie uit ongestructureerde data.

Gestructureerde data

Data afkomstig uit een relationele databank (RDBMS) is meestal gestructureerd. Deze data is meestal ingedeeld in categorieën, denk bijvoorbeeld maar aan postcode, naam, klantennummer, ... Hieruit volgt dat deze data heel gemakkelijk te doorzoeken is.

Semi-gestructureerde data

Data afkomstig uit IoT-apparaten of XML-bestanden zijn meestal semi-gestructureerd. In deze data is een bepaalde structuur te herkennen, maar deze data zit niet in een bepaalde tabel-structuur.

Ongestructureerde data

Deze informatie kan niet gemakkelijk worden opgeslagen in relationele databanken (er kan hiervoor geen primair datatype gebruikt worden, maar bijvoorbeeld wel als een BLOB-type). Denk maar aan Excel-sheets, e-mails, muziek, ... Deze data bevat vaak ook heel nuttige informatie die organisaties graag willen gebruiken (McAfee & Brynjolfsson, 2012). Denk bijvoorbeeld maar aan emails: hoe gelukkig zijn klanten over een bepaald product? Hoeveel mails worden er maandelijks ontvangen met klachten?

2.1.2 Wat is een data warehouse?

De definitie van een data warehouse luidt als volgt: *'een subject-georiënteerde, geïntegreerde, tijd-variante, niet-vluchtige collectie van gegevens die in eerste instantie gebruikt wordt bij organisaties om beslissingen te nemen'* (W. H. Inmon, 2005).

Subject-georiënteerd

Dit begrip slaat op het feit dat een data warehouse gebouwd is met als doel het analyseren van data, niet om transacties op toe te passen. Dit wordt uitgebreid besproken in subsectie 2.1.5.

Geïntegreerd

Dit betekent dat de data warehouse een 'centrale' databank is die gegevens bevat vanuit verschillende bronsystemen (bijvoorbeeld gegevens uit het klantenbestand en gegevens uit het verkoopsysteem).

Tijd-variant

Alle data van het verleden, moet terug te vinden zijn in de data warehouse. Dit betekent dat data uit het verleden (bijvoorbeeld een vorige factuur van een klant) moet beschikbaar zijn, ook al is de data in het transactioneel systeem aangepast. Dit wordt toegepast op transactionele data, minder op master data.

Niet-vluchtig

De data die in het systeem zit, moet onveranderlijk zijn, ook al is de data foutief. Om de foutieve data toch aan te passen, zal er een nieuwe rij moeten toegevoegd worden die de juiste data bevat, die een hogere versie bevat dan de vorige rij.

Conclusie

We kunnen dus uit de definitie van een data warehouse afleiden dat het een grote databron is die alle (gestructureerde semi-gestructureerde en ongestructureerde) gegevens bevat die een organisatie bezit vanaf het moment dat de data warehouse geïmplementeerd werd tot het heden. Op deze databron worden dan analyses gemaakt.

2.1.3 Waarom is er nood aan een data warehouse?

Een organisatie heeft tegenwoordig heel wat data ter beschikking. Vaak is deze data gefragmenteerd over verschillende systemen. Wanneer men een analyse wil maken op basis van de verspreide data, zal dat niet evident zijn.

Om deze reden wordt een data warehouse ontworpen. Hierin worden gegevens, verspreid over meerdere bronnen, in één centrale plaats verzameld. Zo kunnen rapporteringen makkelijk en flexibel opgebouwd worden.

Een andere reden voor het opbouwen van een data warehouse is dat je de historiek van alle data kan bijhouden. Wanneer er bijvoorbeeld gegevens aangepast zijn in het transactionele

systeem, dan zijn de oude gegevens vaak moeilijk te achterhalen (omdat deze dan vaak overschreven wordt). Door verschillende versies bij te houden van entiteiten, kan je oudere data makkelijk opzoeken.

Wanneer men rapporteringen wil opvragen aan het transactionele systeem, vergt dit ook extra belasting van de server. Dit komt doordat dat datamodel opgebouwd werd niet geoptimaliseerd is om zware SELECT-queries af te handelen. Dit zou niet alleen de server (van het transactionele systeem) meer belasten, bovendien zal dit ook zorgen voor een tragere rapportering. Indien de server te veel rapporteringen zou moeten opvragen, zou dit kunnen leiden tot een overbelasting waarbij transacties niet meer mogelijk zouden kunnen zijn.

2.1.4 Wat is het doel van een data warehouse?

Het belangrijkste doel van een data warehouse is om een **correcte** rapportering te leveren. Dit zorgt ervoor dat het beslissingsnemingsproces van het management ondersteund wordt.

Data kwaliteit

Zoals eerder aangekaart, is het belangrijk dat rapporten de juiste gegevens bevat. Hieruit volgt dat data kwaliteit een heel belangrijk aspect is. Vaak zijn er verschillende oorzaken waarom de data kwaliteit niet voldoet:

- Inconsistente data tussen verschillende systemen
- Incorrecte gegevens
- Onvoldoende validatie bij het invoeren van gegevens
- Onjuiste gegevensbewerkingen
- ...

(Helfert, Zellner & Sousa, 2002)

Gegevens die in een data warehouse geladen worden, ondergaan een proces (zie paragraaf 2.1.6). In dit proces wordt de data gemanipuleerd zodat de data kwaliteit verhoogd wordt.

Performantie

We kunnen een onderscheid maken tussen 3 verschillende soorten beslissingen: operationele (dagelijks), tactische (jaarlijks) en strategische (lange termijn) beslissingen. Wanneer we operationele rapporten nodig hebben, verwachten we dan ook dat deze onmiddellijk kunnen opgeleverd worden. Het datamodel van een data warehouse wordt geoptimaliseerd voor het ophalen van data in plaats van het te kunnen stockeren. De data wordt 's nachts ingeladen zodat werknemers geen performantie problemen hieromtrent ondervinden.

De toekomst Door de komst van in-memory databanken merken we op dat een deel van de (operationele) rapportering opnieuw verhuist naar de transactionele databanken. Dit heeft enerzijds te maken met de snelheid van de databanken en anderzijds met het feit dat queries om operationele rapporten op te vragen gebruikelijk niet zo belastend zijn. Zo kan er gewerkt worden met **live data** (doordat deze niet 's nachts moet ingeladen worden in de data warehouse). De voorwaarde hiervoor is dat alle benodigde data beschikbaar is binnen dat geïntegreerd systeem. Een voorbeeld van een in-memory databank is HANA, een technologie ontwikkeld door SAP.

Automatisering

Doordat alle rapporteringsnoden geautomatiseerd kunnen worden, heeft dit natuurlijk als voordeel dat personeelsleden deze niet meer manueel hoeven te maken/berekenen. Zo kunnen ze hun tijd spenderen aan andere prioriteiten. Deze data kan dan worden voorgesteld in een overzichtelijke omgeving (zie sectie 2.4). Bovendien zal er ook een kleinere kans zijn op fouten.

2.1.5 Wat is OLTP en wat zijn de verschillen met OLAP?

On-line transactional processing (OLTP) systemen zijn voornamelijk klantgericht. Het datamodel is opgebouwd rond het efficiënt verwerken van transacties. On-line analytical processing (OLAP) systemen zijn marktgericht. De data in een OLAP-systemen worden gebruikt om analyses op uit te voeren (Satyanarayana, 2010).

Inhoudelijk

Bij OLAP systemen worden metadata opgeslagen bij de entiteiten. Voorbeelden hiervan zijn: tijdstip van inladen, van welke bron de data komen, ... Het grote voordeel hierbij is dat wanneer een fout gebeurt, er gemakkelijker kan achterhaald worden waar het fout liep. Ook wordt de historische data bewaard, in tegenstelling tot OLTP. Bij OLTP wordt de te wijzigen data overschreven. Het gevolg hiervan is dat de volume data bij OLAP doorgaans groter zal zijn.

Toegankelijkheid

Wanneer men data wil verkrijgen/wijzigen in een OLTP systeem, moet er rekening gehouden met een aantal aspecten. Een transactie in een OLTP omgeving moet voldoen aan enkele eisen:

- **Atomic:** Wanneer een transactie afgebroken is, mag er niets gewijzigd zijn in de databank.
- **Consistent:** Als een deel van de transactie faalt, zullen alle doorgevoerde wijzigingen in die transactie ongedaan gemaakt worden en zal de databank terugkeren naar een consistente staat.

- **Isolated:** Transacties worden geïsoleerd, transacties mogen elkaar in geen enkel geval beïnvloeden.
- **Durable:** Wanneer een transactie is doorgevoerd, kan deze niet meer ongedaan gemaakt worden.

Bij een OLAP-systeem worden geen transacties doorgevoerd, enkel leesopdrachten. Dat vermindert de complexiteit en verhoogt de snelheid van de queries (Satyanarayana, 2010).

2.1.6 Wat zijn de benodigdheden voor een data warehouse?

Voor er kan begonnen worden met het opbouwen van een data warehouse, zijn er enkele benodigdheden. Zo zal er een keuze moeten gemaakt worden voor een bepaalde methodologie en een architectuur. Ook zal er een fysieke opslagplaats nodig zijn. Hiervoor kan gebruik gemaakt worden van Cloud oplossingen of een on-premise server. Maar in dit hoofdstuk bespreken we welke software-aspecten er nodig zijn bij het opbouwen van de data warehouse.

RDBMS

Voor het opmaken en beheren van de data warehouse zal er een RDBMS moeten gekozen worden. Hiervoor zijn heel wat mogelijkheden beschikbaar op de markt. Bijvoorbeeld:

- Oracle DB
- Microsoft SQL Server
- IBM DB2
- Microsoft Office Access
- ...

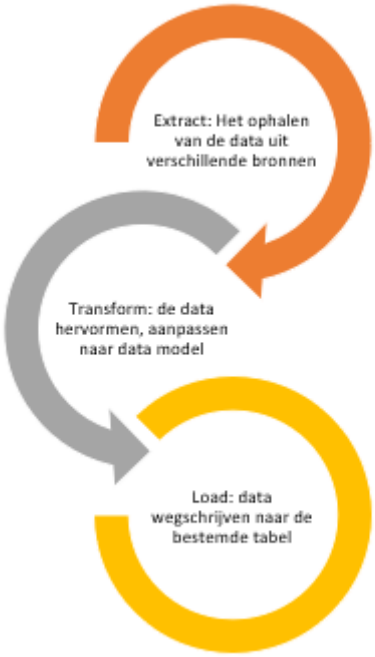
Klassieke databanken bewaren hun gegevens op harde schijven en/of SSD's. Maar sinds kort zien we de komst van een nieuwe technologie, genaamd een in-memory databank. Hierbij worden de gegevens initieel opgeslagen in het RAM-geheugen. Dit zorgt voor een veel snellere lees- en schrijftijd. Het nadeel van deze nieuwe technologie is het prijskaartje.

Data integratie software

Het inladen van data is een proces dat verantwoordelijkheid draagt voor de data integratie software, al is dat niet zijn enige verantwoordelijkheid. De software is verantwoordelijk voor het gehele ETL-proces (zie figuur 2.1):

- Extraction: Ophalen van de data vanuit de bron.
- Transformation: Transformaties en manipulaties uitvoeren op die data.
- Load: De data wegschrijven naar de nieuwe bron.

Voorbeelden van ETL-tools die beschikbaar zijn op de markt zijn: Microsoft SSIS, SAP HANA SDI, Oracle Data Integrator, IBM InfoSphere DataStage, ...



Figuur 2.1: Het ETL-proces (gemaakt via Microsoft Word).

2.2 Dimensioneel modelleren

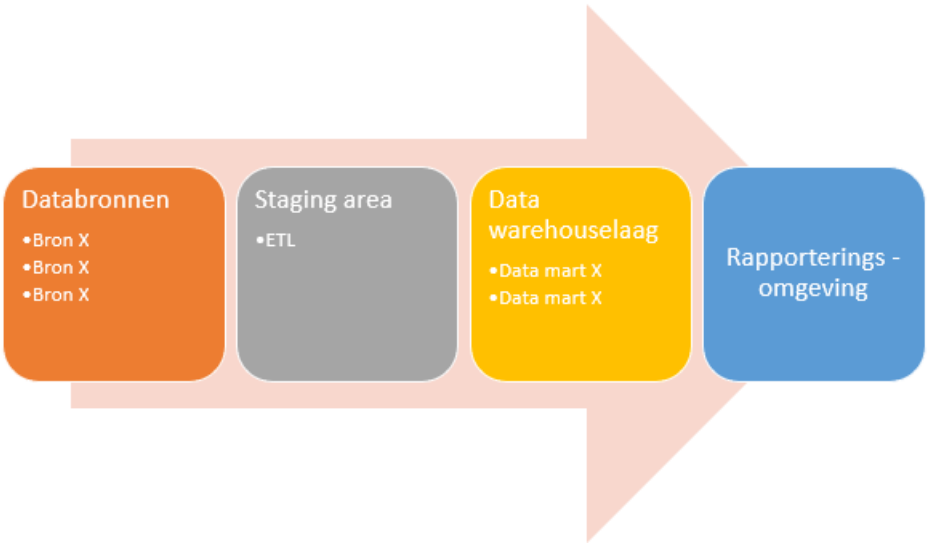
Wie consultant Business Intelligence is, heeft ongetwijfeld al gehoord van dimensioneel modelleren. Over de jaren heen is het als het ware een standaard geworden wanneer men een data warehouse wil ontwerpen. Dimensioneel modelleren heeft als voordeel dat het model niet complex is, dus gemakkelijk te begrijpen, zelf voor niet IT-opgeleide personen. Ook is er vaak een snel resultaat beschikbaar. In deze sectie zal het dimensioneel modelleren aan de hand van Kimball dieper bekeken worden.

2.2.1 Architectuur

Bij het dimensioneel modelleren wordt het proces opgedeeld in 2 soorten lagen: de staging area en de data warehouselaag (voorgesteld in figuur 2.2). In de data warehouselaag worden de data marts opgebouwd. Rapporteringsomgevingen connecteren met de data marts om hun data op te halen.

Staging area

In deze laag wordt het ETL-proces toegepast. Eerst en vooral worden de data vanuit één of meerdere bronnen in de data warehouse geladen. Daarna wordt de data bewerkt en gemanipuleerd. Bijvoorbeeld records met bepaalde lege waarden weglaten. De data kwaliteit (zoals eerder aangekaart) is zeer belangrijk in een data warehouse. Het doel is om de opgehaalde data consistent te maken en ervoor te zorgen dat de integriteit gewaarborgd



Figuur 2.2: Architectuur van een data warehouse op basis van het dimensioneel modelleren (gemaakt via Microsoft Word).

blijft (Kimball & Ross, 2013)

Data warehouselaag

In deze laag worden OLAP-cubes of relationele ster schema's gemaakt op basis van de staging area. Deze laag kan eigenlijk als de presentatielaag beschouwd worden. Deze laag moet gedetailleerde data bevatten. Een data mart moet gebaseerd zijn rond een business unit (Kimball & Ross, 2013).

2.2.2 Componenten

Een sterschema of OLAP-cube bestaat uit dimensies en facts. Wat deze precies zijn, wordt hieronder uitgelegd.

Dimension tabel

Voor elke dimensie wordt een primary key aangemaakt (of afkomstig uit het systeem als business sleutel). Deze sleutel wordt gebruikt in een fact tabel als foreign key, zodat er een relatie kan worden gelegd tussen beide attributen.

Naast de primary key wordt in deze tabel ook de beschrijvende data bewaard voor een bepaalde rij. Deze data kunnen gebruikt worden om in de rapporteringsomgeving de verschillende assen te kiezen. Een typisch attribuut is bijvoorbeeld 'naam' of 'woonplaats' voor de dimensie 'Customer'. Deze data wordt gebruikt om de transactionele data die in de fact tabel zit te beschrijven.

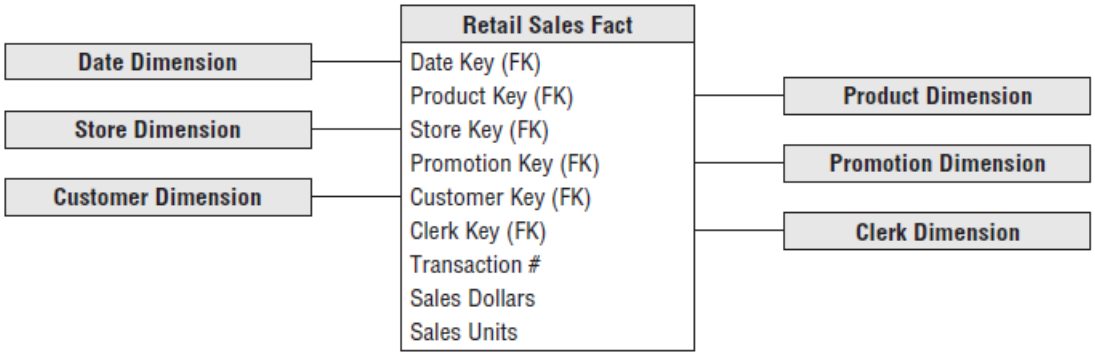
Fact tabel

In een fact tabel worden alle meetbare cijfers bijgehouden. Meetbare cijfers betekent dat bewerkingen moeten mogelijk zijn op die data. Een bankrekeningnummer is een getal, maar hier kunnen geen bewerkingen met uitgevoerd worden (bijvoorbeeld gemiddelde bankrekeningnummer geven). Facturatiebedrag is een goed voorbeeld. Hierop kunnen enkele bewerkingen worden uitgevoerd, bijvoorbeeld: gemiddelde, minimum, maximum, totaal, ... Deze gegevens worden in vaktermen als **measures** aangeduid.

De fact tabel bevat niet alleen measures, maar ook de foreign keys van de dimensies waarmee het verbonden is. Zo kan je bruikbare informatie toevoegen aan je measure in de rapporteringsomgeving.

Ster schema

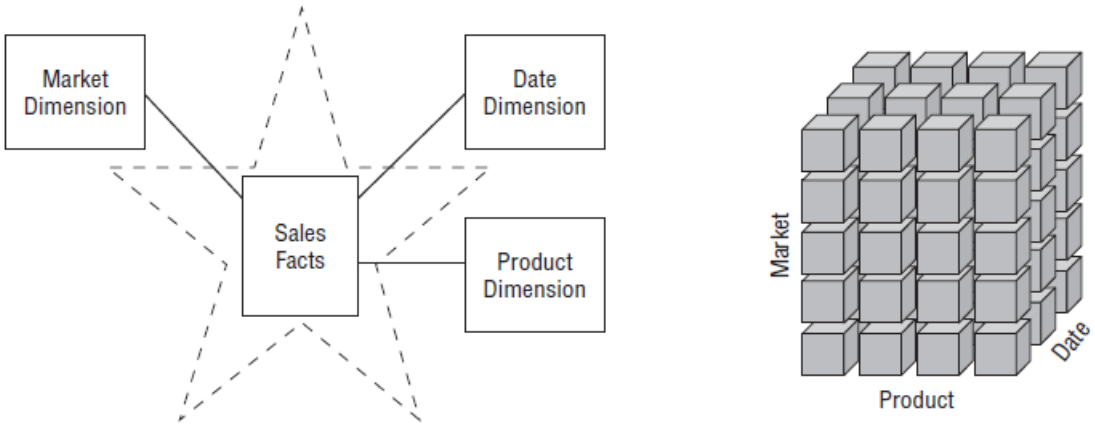
In een ster schema worden dimensies en fact tabellen verbonden door enerzijds de primaire sleutels in de dimensies, en anderzijds bij de vreemde sleutels in de fact tabel. Wanneer meerdere tabellen verbonden worden met elkaar zien we een centraal punt in het model, dat de fact tabel is (zie figuur 2.3).



Figuur 2.3: Ster schema voorgesteld door Kimball en Ross (2013).

Verschil tussen een sterschema en een OLAP-cube

Het verschil tussen beide zit niet in het ontwerp, maar puur in het 'fysieke' gedeelte. OLAP cubes zijn geoptimaliseerd voor een drill down of een drill up te doen in de gegevensset. Drill down betekent dat de gegevens op een dieper detailniveau zullen bekeken worden, bijvoorbeeld vertrekkend uit een productcategorie niveau, ga je naar een productniveau. OLAP cubes zorgen ervoor dat er meer analytische functies beschikbaar zijn in vergelijking met SQL. Maar als nadeel heeft de OLAP cubes dat het niet zo performant is als een ster schema (zie figuur 2.4). (Kimball & Ross, 2013).



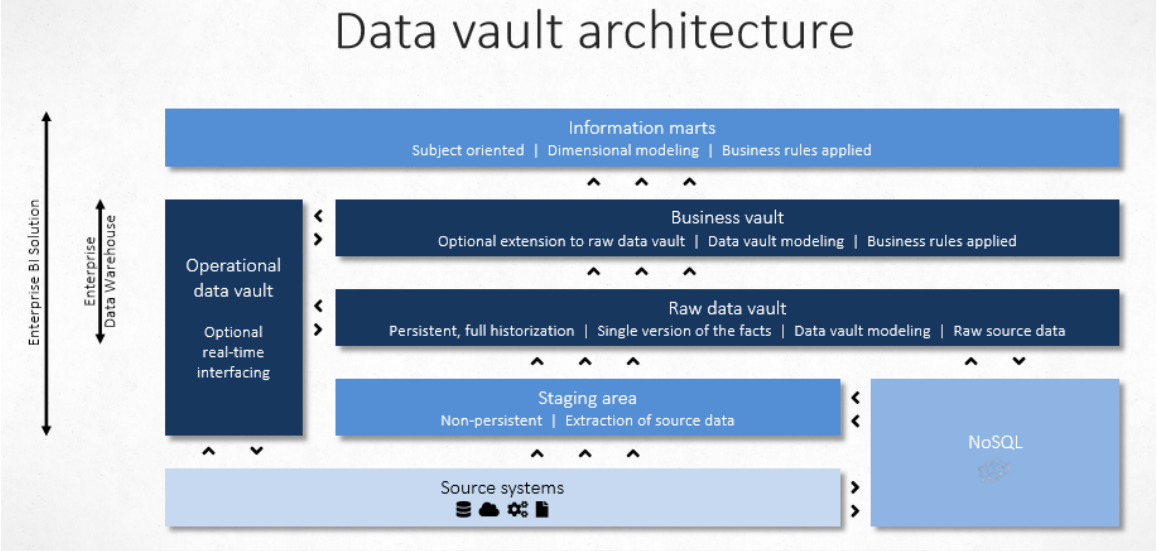
Figuur 2.4: Sterschema (links) en OLAP cube (rechts) voorgesteld door Kimball en Ross (2013).

2.3 Modelleren via Data Vault 2.0

Data Vault 2.0 is een modelleertechniek die ontworpen is door Daniel Linstedt. Het model zorgt ervoor dat dimensies gemakkelijk uitgebreid kunnen worden en dat databronnen toevoegen vlot moet gaan. Linstedt is van mening dat business requirements vaak veranderen, dus moet het model waarin de data warehouse ontworpen is ook flexibel zijn (D. Linstedt & Olschimke, 2016).

2.3.1 Architectuur

Data vault maakt gebruik van een 3-lagen model (zie figuur 2.5). Dit heeft als voordeel dat processen duidelijk kunnen onderscheiden worden per laag en alles overzichtelijk blijft. Deze architectuur ondersteunt om data op halen via een batch, maar ook om live data op te halen. NoSQL kan ook gebruikt worden om de data warehouse te ontwerpen. Wanneer de data live opgehaald wordt, valt de staging area weg en wordt de data onmiddellijk in de raw data vault geladen (Krneta, Jovanović & Marjanović, 2014).



Figuur 2.5: Data Vault architectuur voorgesteld door Stroobants (2018).

Staging area

In deze laag wordt alle data ingeladen (of een virtuele tabel gebruikt) van een bepaalde bron via een batch. Hier wordt alle data onbewerkt ingeladen. Deze data bevat dan ook nog geen historische metadata. De tabel worden dus gedupliceerd van de bron(nen). Deze laag is niet persistent.

Raw data vault

De data worden overgeladen van de staging area naar de raw data vault via het ETL-proces (2.1.6). Deze laag is persistent, logisch ook wanneer we verschillende versies en de

historiek behouden van onze entiteiten. Vanaf deze laag beginnen we te modelleren in Data Vault. De data wordt in deze laag gemanipuleerd (ofwel getransformeerd (ETL)). Ook wordt er metadata toegevoegd aan de records zodat er audits kunnen plaatsvinden.

Business vault

Dit is een optionele laag. Deze laag wordt enkel en alleen toegevoegd wanneer er 'Business regels' moet toegepast worden in het model. De business vault wordt in principe niet opgeslagen in een aparte laag, maar vaak wordt deze opgeslagen als een uitbreiding van de raw data vault. Een voorbeeld van een 'Business rule' is dat je bijvoorbeeld geen producten wil promoten wanneer er minder dan 10 in voorraad zijn.

Information marts

Vertrekkend uit de business vault (of raw data vault) zullen er information marts moeten aangemaakt worden, ook wel bekend als data marts . D. Linstedt en Olschimke (2016) spreek liever over 'information' mart omdat het doel van een enterprise data warehouse duidelijk is: informatie aanbieden. De information marts bestaan uit sterschema's. Hierop connecteren de eindgebruikers om hun informatie te verkrijgen.

2.3.2 Componenten

Een data vault model bestaat uit 3 soorten componenten: hubs, links en sattelites. Elk component heeft zijn functie en zijn doel.

Hubs

D. Linstedt en Olschimke (2016) beschrijft hubs als pilaren voor het Data Vault model. Een hub bestaat altijd uit minimaal 4 attributen:

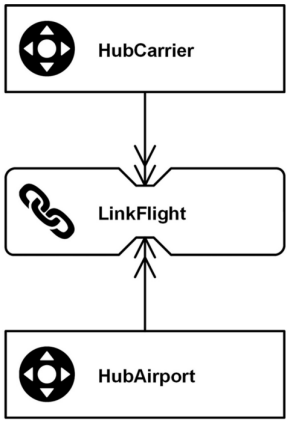
- **Hashkey (PK):** Als primary key van de entiteit wordt een gehashte identifier samen met de naam van de bron van de entiteit gebruikt.
- **LoadDate:** Datum/tijdstip wanneer de record geëxtraheerd werd uit de bron.
- **Record source:** Van welke databron is de record afkomstig?
- **Business key(s):** De business key(s) van de bijhorende entiteit

De business key is een unieke sleutel, die vaak een betekenis heeft naar de business. Voorbeelden zijn: ISBN-nummers, Klantnummer, Chassisnummer, ...

Hubs zijn vooral heel handig wanneer er meerdere databronnen zijn, zo kan je meerdere bronnen aan een hub hangen. In een hubs zit nooit andere data behalve een hash key, metadata en business keys.

Links

Wanneer we 2 hub-entiteiten willen verbinden (zie figuur 2.6), zullen we ze niet rechtstreeks met elkaar verbinden. Twee hub-entiteiten worden namelijk verbonden door middel van een link. Andere verantwoordelijkheden voor een link zijn hiërarchieën, redefinities of business termen. De bedoeling is om een zo laag mogelijke granulariteit te creëren. Links zorgen ervoor dat het data vault model heel flexibel wordt en makkelijk uitbreidbaar is.



Figuur 2.6: Link entiteit die 2 hub entiteiten met elkaar verbindt. (D. Linstedt & Olschimke, 2016).

Ook bij links moeten er een aantal attributen aanwezig zijn:

- **Hashkey (PK):** Als primaire sleutel van de entiteit worden alle business keys gehasht naar 1 sleutel.
- **LoadDate:** Datum/tijdstip wanneer de record geëxtraheerd werd uit de bron.
- **Record source:** Van welke databron is de record afkomstig?
- **Business key(s):** Alle business key(s) van de 2 gelinkte hub-entiteiten.

Sattelites

In een satellite worden alle gegevens gestockeerd die een business object, relatie of transactie beschrijven. In de entiteit zelf is het belangrijk dat de historiek wordt bijgehouden (D. Linstedt & Olschimke, 2016).

Bij een satellite vinden we minstens volgende attributen terug:

- **Parent Hashkey (PK):** Als primaire sleutel van de entiteit worden alle business keys gehasht naar 1 sleutel samen met de naam van afkomst.
- **LoadDate (PK):** Datum/tijdstip wanneer de record geëxtraheerd werd uit de bron.
- **Record source:** Van welke databron is de record afkomstig?
- **End load date:** Hierin wordt het moment geladen wanneer de entiteit niet meer gebruikt wordt (belangrijk voor het bewaren van de historiek). Wie vertrouwd is met het dimensioneel modelleren, kan dit vergelijken met het principe van slow changing dimensions.

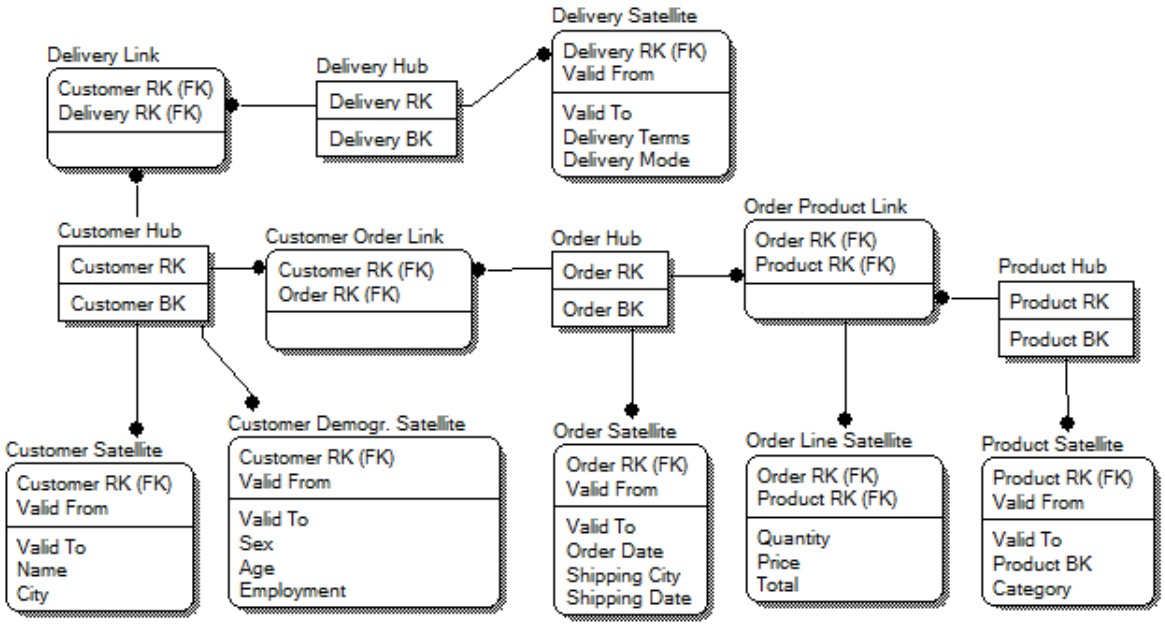
- **Hash difference:** Hierin worden alle dimensionele data samen gehasht naar 1 sleutel. Zo kan er makkelijk vergeleken worden of de nieuwe data wel degelijk een verandering is of niet.

Een satellite hoort bij een hub of een link. Een hub en een satellite vormen een bepaald business object. Hier is de primary key een composite key, dat bestaat uit een Parent Hashkey en een LoadDate. Deze combinatie is nodig om nieuwe versies van data te kunnen toevoegen.

Data Vault schema

Wanneer we zowel hubs, links als sattelites samengieten in één schema, bekomen we een Data Vault (zoals in figuur 2.7). Wel zijn er nog enkele belangrijke opmerkingen:

- Hubs mogen nooit rechtstreeks verbonden met elkaar, dit moet altijd gebeuren via een link (anders verliest het model zijn flexibiliteit).
- Een link kan meer dan 2 hubs verbinden.
- Sattelites kunnen zowel met hubs als links verbonden worden.
- Hubs/links kunnen meerdere sattelites hebben: deze staan meestal voor verschillende databronnen.
- Een satellite kan maar verbonden worden met 1 hub of link.



Figuur 2.7: Een voorbeeld van een Data Vault model (Bukhantsov.org)

Dit model heeft veel flexibiliteit te bieden. Enerzijds kan men gemakkelijk nieuwe hubs toevoegen door een nieuwe link te leggen. Ook kan men heel gemakkelijk een nieuwe gegevensbron toevoegen, dit wordt gedaan door een nieuwe satellite toe te voegen aan een bestaande hub.

2.4 Rapporteringsomgevingen

Wanneer alle data in de data warehouse ingeladen en getransformeerd is, moet het mogelijk zijn om visuele en interactieve rapporten op te stellen. Op basis van deze rapporten kan het management beslissingen gaan nemen. Ook kunnen dashboards opgesteld worden voor werknemers die een high level overzicht nodig hebben. De bedoeling in deze sectie is om een schets te geven van de rapporteringsomgevingen die op de markt beschikbaar zijn en wat de betekenis is van een KPI.

2.4.1 Wat is een KPI?

Een Key Performance Indicator (KPI) is een doelstelling uitgedrukt als een cijfer voor een bepaalde actie van een bedrijf. Een KPI is niet sector gebonden, maar wordt bepaald door het management van een onderneming. Het is een doelstelling die het bedrijf moet verwezenlijken. Vaak worden deze doelstellingen op lange termijn bepaald.

Een KPI moet opgesteld zijn aan de hand van het SMART-principe:

- **Specifiek:** Een KPI moet duidelijk geformuleerd worden.
- **Meetbaar:** Er moet kunnen vastgesteld worden wanneer een doel bereikt is.
- **Acceptabel:** Is de KPI haalbaar?
- **Realistisch:** Is het behalen van deze KPI realistisch?
- **Tijdsgebonden:** Wanneer moet de KPI behaald worden?

Alle KPI's samen vormen de rapporteringsnoden op strategisch niveau.

2.4.2 Het magische kwadrant

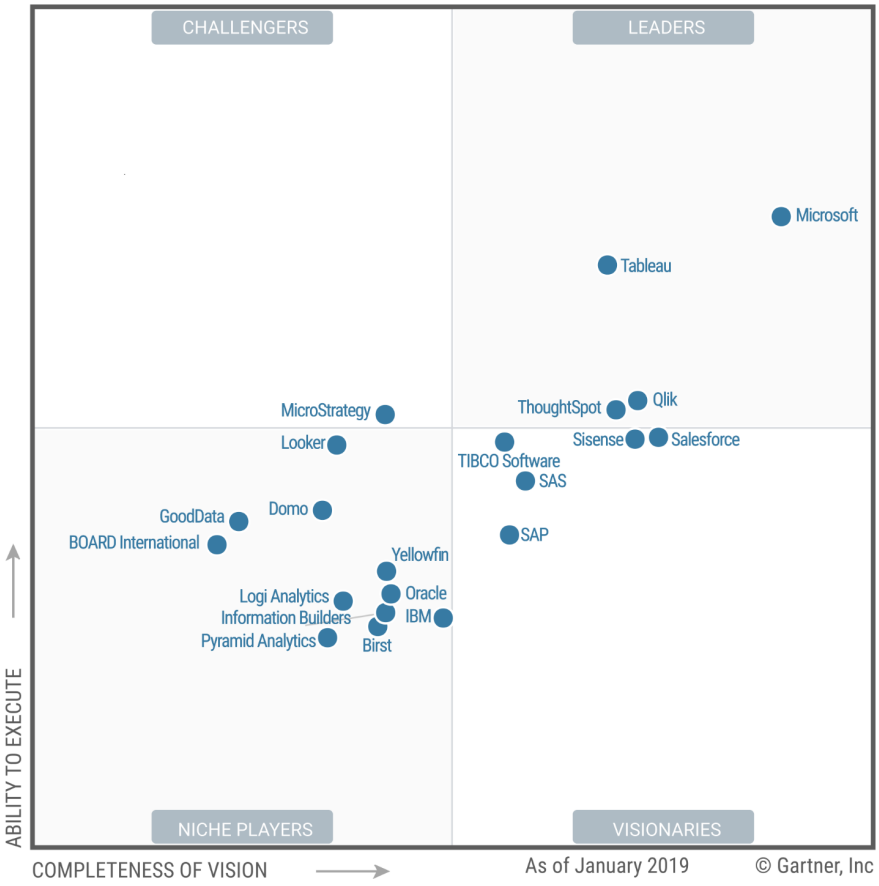
Gartner (2019) maakte een vergelijkende studie tussen verschillende Business Intelligence platvormen. Als meetpunt werd in deze studie rekening gehouden met 15 cruciale features:

- **Administratie, beveiliging en architectuur:** administratie gebruikers, beveiliging garanderen, controleren wie toegang heeft en indien nodig, een herstel kunnen uitvoeren.
- **Platform-as-a-service:** heeft de software een omgeving beschikbaar in de cloud?
- **Connecteren naar databronnen:** kunnen gebruikers verbinden naar verschillende databronnen (ongestructureerde data, gestructureerde data, ..)?
- **Beheer van metadata:** mogelijkheid aanbieden om te kunnen werken met metadata (zoeken, opslaan, herstellen, ...).
- **Opslag en laden:** mogelijkheden voor het integreren, transformeren en het laden van gegevens.
- **Voorbereiden van gegevens:** op welke manieren is het voorbereiden van gegevens mogelijk? Kan er gebruik gemaakt worden van machine learning?
- **Schaalbaarheid:** hoe wordt er omgegaan met enorme volumes data, complexe datamodellen, zijn de prestaties geoptimaliseerd, en hoe gaat de omgeving om met

veel gebruikers?

- **Geavanceerde analyses voor data scientists:** Zijn de geavanceerde analytische mogelijkheden makkelijk beschikbaar of moet er externe ontwikkelde software geïmplementeerd worden?
- **Dashboards:** De mogelijkheid om interactieve dashboards te creëren, is werken met geo-informatie mogelijk?
- **Verkennen:** kunnen gebruikers data analyseren en manipuleren door te werken met een interactieve presentatie van die gegevens?
- **Augmented data ontdekken:** kan data ontdekt worden door natural language query (NLQ) technologie? Kan het automatisch uitzonderingen, clusters, links en voorspellingen vinden en visualiseren?
- **Invoeren van Analytics in externe applicaties:** kunnen deze dashboards of visualiseringen makkelijk in een andere omgeving worden ingevoerd?
- **Publiceren, delen en samenwerken:** kunnen gebruikers de inhoud publiceren, delen of samenwerken met anderen om visualisaties te realiseren of bekijken?
- **Gebruiksgemak, visuele aantrekkelijkheid en de integratie van de workflow:** is het platvorm makkelijk te gebruiken en beheren? Kan de data visueel aantrekkelijk voorgesteld worden? Kan de tool makkelijk geïmplementeerd worden in huidige workflow?

Op basis van deze features werden de verschillende Business Intelligence platvormen verdeeld in vier kwadranten naargelang hun visie en naar hoelang ze deze realiseren (zie figuur 2.8). De vier kwadranten zijn: niche-spelers, uitdagers, visionairs en marktleiders.



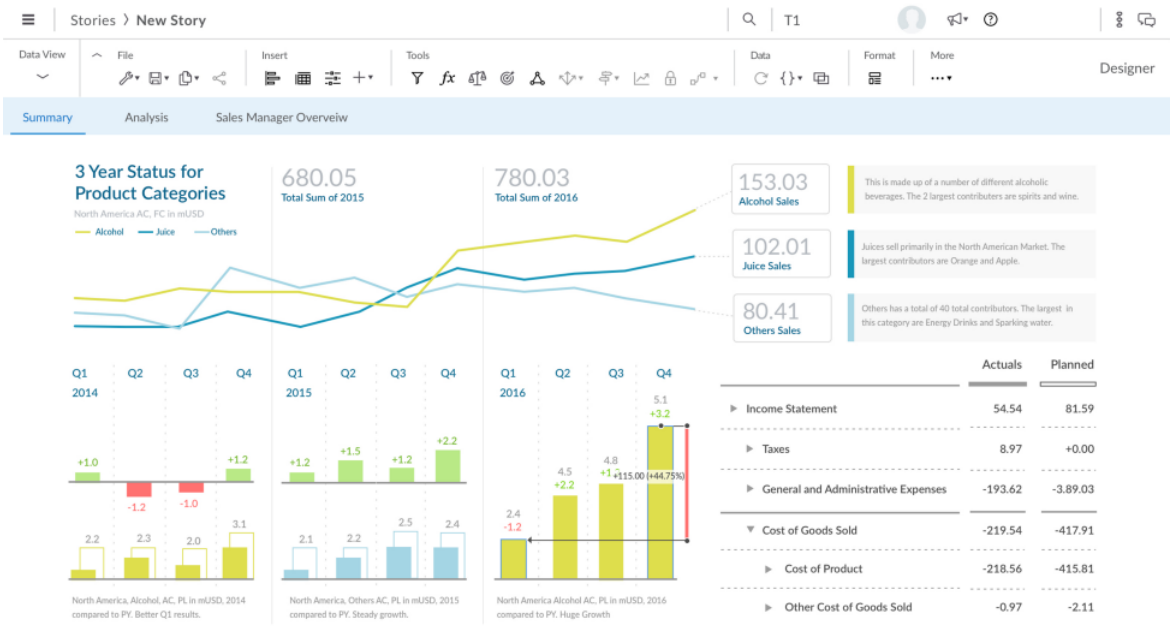
Figuur 2.8: Het magische kwadrant over Selfservice BI opgesteld door Gartner (2019).

In deze literatuurstudie bekijken we enkel de mogelijkheden die 2 grootmachten in de ERP-markt (Microsoft en SAP) aanbieden, maar er zijn dus nog een aantal andere opties beschikbaar om rapporteringen visueel aantrekkelijk te maken.

2.4.3 SAP Analytics Cloud

Bij SAP Analytics Cloud kan er gebruik gemaakt worden van realtime analytics. Hiervoor is live data nodig en deze wordt opgehaald in het transactionele systeem. Indien gewenst, kan er toch nog steeds gewerkt worden met een batch die 's nachts geladen wordt.

Analytics cloud biedt heel veel mogelijkheden om interactieve dashboards te ontwerpen (zoals in figuur 2.9). Waar SAP Analytics Cloud zich onderscheidt met de andere spelers, is dat er in deze omgeving planning kan worden toegepast. Er kunnen budgetten opgesteld worden voor de komende jaren (bijvoorbeeld IT-kosten), en deze kunnen dan vergeleken worden met de actuele kosten op dat moment.

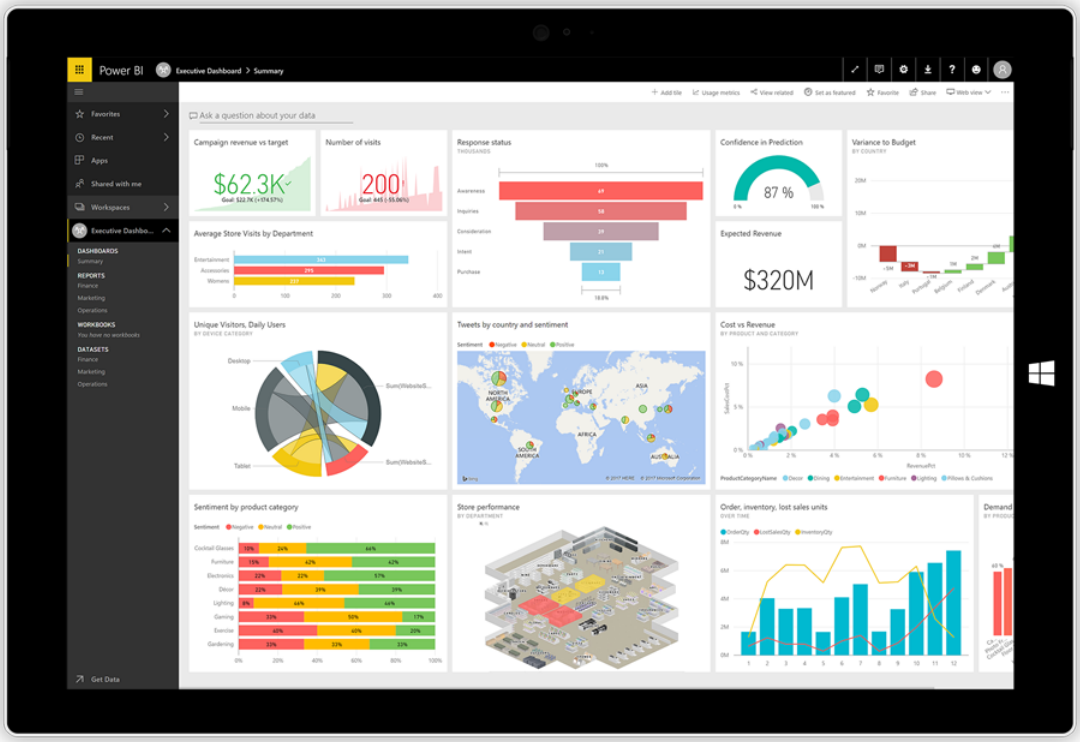


Figuur 2.9: Een voorbeeld van een dashboard gemaakt met SAP Analytics Cloud (sap.com)

2.4.4 Power BI

Power BI is het programma bij uitstek dat gebruikt wordt bij datavisualisatie (voorbeeld in figuur 2.10) van ontwikkelaar Microsoft. PowerBI is een uitstekende keuze wanneer bedrijven Office 365 en Microsoft Dynamics geïmplementeerd hebben in hun infrastructuur.

In tegenstelling tot SAP Analytics cloud kan je Power BI wel on-premise installeren. Maar indien gewenst, kan er steeds gebruik gemaakt worden van een cloud-omgeving.



Figuur 2.10: Een voorbeeld van een dashboard gemaakt met Power BI (Microsoft.com)

3. Opzet van het onderzoek

Voor er aan de slag kan gegaan worden met het onderzoek, moet er een data warehouse opgesteld worden zowel via de Data Vault methodologie als de methodologie voor het dimensioneel modelleren. In dit hoofdstuk wordt de opbouw van het experiment uitgebreid uitgeschreven. Bovendien wordt er ook een overzicht gegeven over de rapporteringsnood voor DHL Pharma Logistics, wat de betekenissen zijn van de gebruikte data in dit onderzoek en hoe een remote connectie moet opgezet worden voor SAP HANA.

3.1 Rapporteringsnood DHL Pharma Logistics

In deze sectie wordt beschreven wat de rapporteringsnood is voor DHL Pharma Logistics. Er wordt duidelijk beschreven wat de KPI is en deze wordt onderworpen het SMART-principe.

3.1.1 Context

DHL Pharma Logistics is een divisie van de logistieke grootmacht DHL Supply Chain. PHL Pharma Logistics is verantwoordelijk van de opslag en het bewaren van allerlei medische producten. Bij het stockeren van medische producten moet er met allerlei zaken rekening gehouden worden: temperatuur, houdbaarheid, ... De verantwoordelijkheid van DHL Pharma Logistics ligt bij de opslag en niet bij het vervoeren van deze middelen. DHL Pharma Logistics ontvangt de goederen van labo's (ontwikkelaars van de medicatie) en houden deze goederen bij tot deze moeten verzonden worden naar de klant (ziekenhuizen, apothekers, ...).

3.1.2 Dock-to-Stock proces

De verantwoordelijkheid van DHL Pharma Logistics is om de toegekomen goederen van een pharmaceutische klant te stockeren op de juiste plaats in een vooraf bepaalde tijdspanne. Deze periode wordt opgenomen in een Service Level Agreement (SLA). Het berekenen van deze KPI kan op verschillende niveau's: levering, per pallet of per unit. Bij DHL Pharma Logistics is het gebruikelijk dat goederen moeten worden gestockeerd binnen de 24 uur. Deze KPI wordt dan berekend op pallet-niveau, al zijn er enkele uitzonderingen.

Dock-to-Stock onderworpen aan het SMART-principe

Goederen die toekomen aan het magazijn moeten binnen de 24 uur gestockeerd worden op de juiste plaats. In dit interval, moeten alle controles uitgevoerd zijn. Dit is een afspraak die vast gelegd is in de Service Level Agreement met de Labo's.

- **Specifiek:** De KPI is duidelijk geformuleerd.
- **Meetbaar:** Het doel is bereikt wanneer de goederen tijdig zijn gestockeerd.
- **Acceptabel:** De KPI is bepaald in een overeenkomst, dus is deze acceptabel.
- **Realistisch:** Het stockeren van de goederen binnen de 24 uur is realistisch.
- **Tijdsgebonden:** Er wordt een duidelijk interval aangegeven (binnen de 24 uur).

3.1.3 De formule voor het berekenen van de KPI

Een pallet is tijdig gestockeerd wanneer:

$$\text{Tijdstip van stockage pallet} - \text{Tijdstip van aankomst pallet} < 24 \text{ uur}$$

3.1.4 Hoe moet deze KPI bekeken kunnen worden?

Deze opgestelde KPI is niet alleen belangrijk voor het cliënteel om na te gaan of de goederen wel tijdig gestockeerd zijn, maar ook voor DHL Pharma Logistics om de juiste analyse te kunnen maken wanneer het fout loopt. Zo kunnen ze opsporen waar er een probleem zit in hun proces en daar de juiste oplossing voor vinden. Hebben ze te weinig personeel voor het verwerken van de orders? Zijn er veel defecten in hun rollend materieel? Hebben ze te veel leveringen geaccepteerd op een te korte termijn? Dit zijn nog maar enkele vragen die kunnen beantwoord worden wanneer een data warehouse is opgesteld voor deze specifieke KPI.

3.1.5 Benodigde data

De data die nodig is voor het uitwerken van de KPI:

- Een palletnummer en een levernummer
- Tijdstip van levering en stockering.

- Klantengegevens

Alle overige data zijn aanvullingen. Deze kunnen dan gebruikt worden om diepere analyses uit te voeren.

3.2 Betekenis van de brondata

Vooraleer van start gegaan wordt met het onderzoek, volgt in deze sectie een overzicht (tabel 3.1) van wat de betekenissen zijn van de gebruikte data. Zo kan de data goed geïnterpreteerd worden. De data die gebruikt wordt is gegenereerde data, en is niet afkomstig uit een bronsysteem van DHL Pharma Logistics. Merk op: de naamgeving van bepaalde data is aangepast (in samenspraak met de business) zodat deze data makkelijker te begrijpen is door de business. Een primary key wordt aangeduid door PK, een foreign key door FK.

3.2.1 Overzicht betekenissen

Kolomnaam	**Afkomst**	**Betekenis**
LABO_ENTITY_ID (PK)	Customer_entities.csv	Een uniek identificatie nummer voor een labo entiteit. Een labo entiteit is een vestiging van een bepaald bedrijf.
LABO_ENTITY_NAME	Customer_entities.csv	De vestigingsnaam van de entiteit.
LABO_GROUP_ID (FK)	Customer_entities.csv	Identificatienummer van de overkoepelende groep.
LABO_GROUP_ID (PK)	Customer_groups.csv	Identificatienummer van de groep. Een groep bestaat uit meerdere entiteiten.
LABO_GROUP_NAAM	Customer_groups.csv	De naam voor de groep.
STAFF_ID (PK)	Staff.csv	Personeelsnummer van een werknemer.
STAFF_NAME	Staff.csv	De naam van een werknemer.
WAREHOUSE_ID (FK)	Staff.csv	Het warenhuis waar de werknemer actief is.

WAREHOUSE_ID (PK)	Warehouses.csv	Het identificatienummer voor een bepaalde warenhuis.
WAREHOUSE_NAME	Warehouses.csv	De gemeente/stad van het warenhuis (tevens ook de naam).
PRODUCT_ID (PK)	Products.csv	Het artikelnummer. Dit is een uniek nummer.
PRODUCT_DESCRIPTION	Products.csv	Naam/beschrijving van een bepaald product.
LABO_ENTITY_ID (FK)	Products.csv	Het identificatienummer van de labo entiteit waarvan het product afkomstig is.
STATUS_ID (PK)	Dock_to_stock_status.csv	Identificatienummer van een bepaalde status. Een status wordt gegeven aan een pallet wanneer deze gestockeerd wordt. Ofwel is deze op tijd gestockeerd, ofwel is er een bepaalde reden waarom dit niet op tijd kon gestockeerd worden (bijvoorbeeld een te druk moment).
STATUS_DESCRIPTION	Dock_to_stock_status.csv	Beschrijving/uitleg over de status.
LEVERAGE_ID (PK)	Deliveries.csv	Een nummer dat toegekend wordt aan een bepaalde levering. Vaak ook een referentienummer dat de klant (labo entiteit) ziet op de factuur.
PALLET_ID	Deliveries.csv	Het identificatienummer van een pallet. Een levering kan uit meerdere palletten bestaan.
STAFF_ID (FK)	Deliveries.csv	Een personeelsnummer. Dit personeelslid was verantwoordelijk voor het juist en tijdig (binnen de afgesproken periode met de labo entiteit) stockeren van een pallet.

PRODUCT_ID (FK)	Deliveries.csv	Het identificatienummer van het product dat gestockeerd werd. Er wordt enkel 1 productsoort op een pallet gestockeerd.
PRODUCT_QUANTITY	Deliveries.csv	De hoeveelheid producten aanwezig op een pallet. De hoeveelheid wordt weergegeven per unit.
DOCK_TIME	Deliveries.csv	Het tijdstip waarop de pallet toekomt in het magazijn.
STOCK_TIME	Deliveries.csv	Het tijdstip waarop de pallet gestockeerd werd op de juiste plaats.
STATUS_ID (FK)	Deliveries.csv	Identificatienummer voor de status. Indien deze tijdig werd gestockeerd, dan werd geen status toegekend.

Tabel 3.1: Betekenis van de gebruikte data in dit experiment.

3.3 Gebruikte technologieën in dit experiment

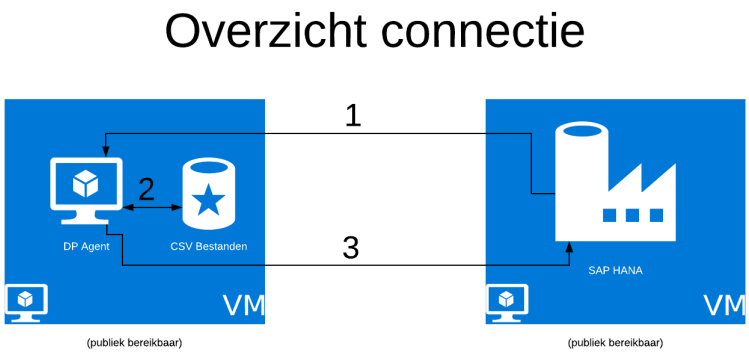
De benodigdheden voor het nabootsen van dit experiment zijn:

- **SAP HANA technologie:** deze databank draait op een Linux distributie (Debian) in een Azure omgeving.[1]
- **Databron:** Als databron worden csv-bestanden gebruikt die aangeleverd worden door een Windows server (versie 2016) die draait in een Azure omgeving.[2]
- **Data Provisioning Agent:** De DP Agent wordt gebruikt om een bron te verbinden met SAP HANA.
- **SDI:** Een ETL-tool die aangeleverd wordt door SAP en die geïntegreerd is binnen SAP HANA.

In dit onderzoek worden voornamelijk SAP producten gebruikt. De reden hiervoor is omdat er binnen DHL Pharma Logistics ook voornamelijk gewerkt wordt met SAP producten.

[1] De SAP HANA server moet bereikbaar zijn over het internet.
[2] De Windows server moet bereikbaar zijn over het internet.

3.4 Overzicht van de connectie tussen SAP HANA en de host



Figuur 3.1: Voorstelling netwerk (gemaakt via Lucidchart.com).

In figuur 3.1 zijn beide virtuele machines bereikbaar op het internet zodat de connectie tussen beiden mogelijk is. Op de virtuele machine waar de DP Agent op draait staat bovendien ook poort 5050 open en maakt gebruik van het TCP protocol. Dit protocol zorgt ervoor dat elk datapakket arriveert op zijn bestemming in tegenstelling tot UDP. Wanneer snelheid belangrijk (Skype, streaming, ..) is, kies je voor UDP. Dit protocol zal nooit gebruikt worden bij deze soort connectie.

Hieronder een overzicht over hoe de verbinding in zijn werk gaat:

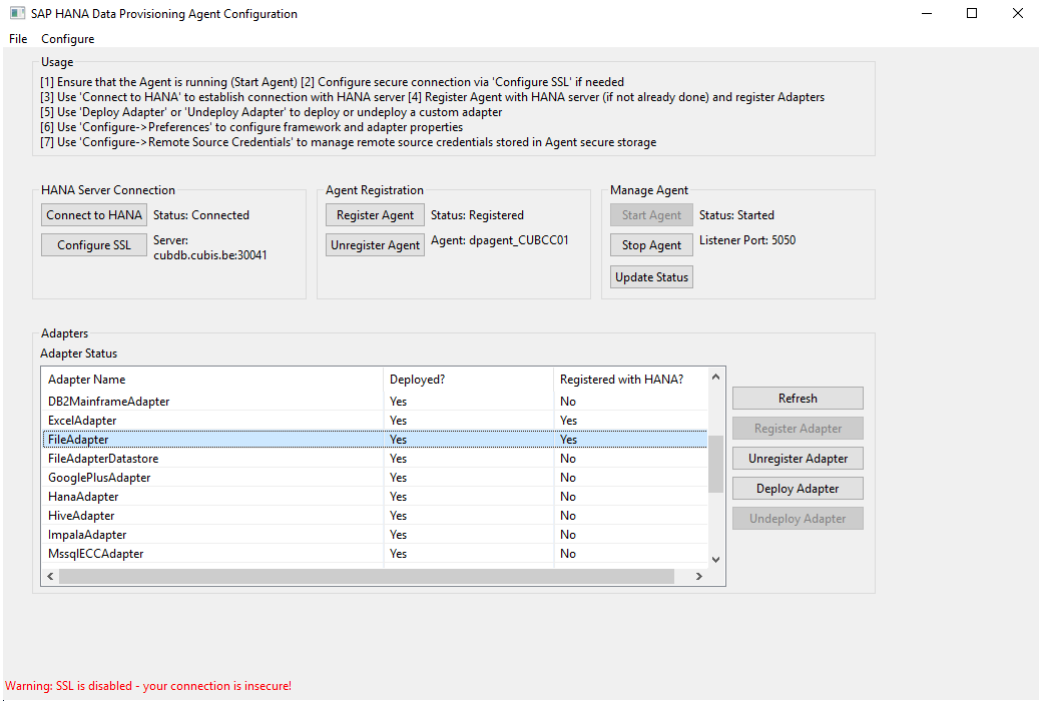
1. SAP HANA stuurt een request voor data naar de DP Agent die geïnstalleerd staat op de host.
2. De host haalt de benodigde data op via een adapter, in dit geval haalt hij de CSV-files die nodig zijn op uit een lokale betandslocatie.
3. De DP Agent zendt de data door naar SAP HANA.

3.5 Opzetten van een remote source

Wanneer er een verbinding moet opgezet worden naar SAP HANA, moet een Data Provisioning Agent (zie figuur 3.2) geïnstalleerd worden. Deze wordt geïnstalleerd op de host die een verbinding moet maken met SAP HANA. Nadat deze geïnstalleerd is, moet deze ook nog geconfigureerd worden. Volgende zaken moeten zeker in orde gebracht worden voor een connectie kan plaats vinden:

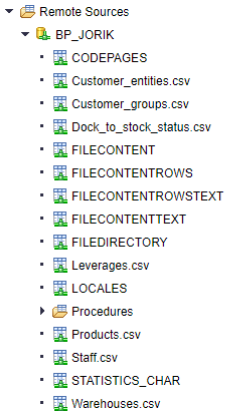
- Connection: de agent moet geconnecteerd en ingelogd zijn op het SAP HANA systeem.
- Registered: de agent moet geregistreerd zijn bij SAP HANA.
- Adapter: de juiste adapter (in dit onderzoek: FileAdapter) moet geregistreerd zijn

bij SAP HANA[1]. Bovendien moet de adapter ook correct geconfigureerd zijn (juiste parameters).



Figuur 3.2: DP Agent verbonden met SAP HANA.

Wanneer alles correct geconfigureerd is, kan er nu vanuit SAP HANA een remote connectie (zie figuur 3.3) gemaakt worden naar de host en kan de benodigde informatie opgevraagd worden.



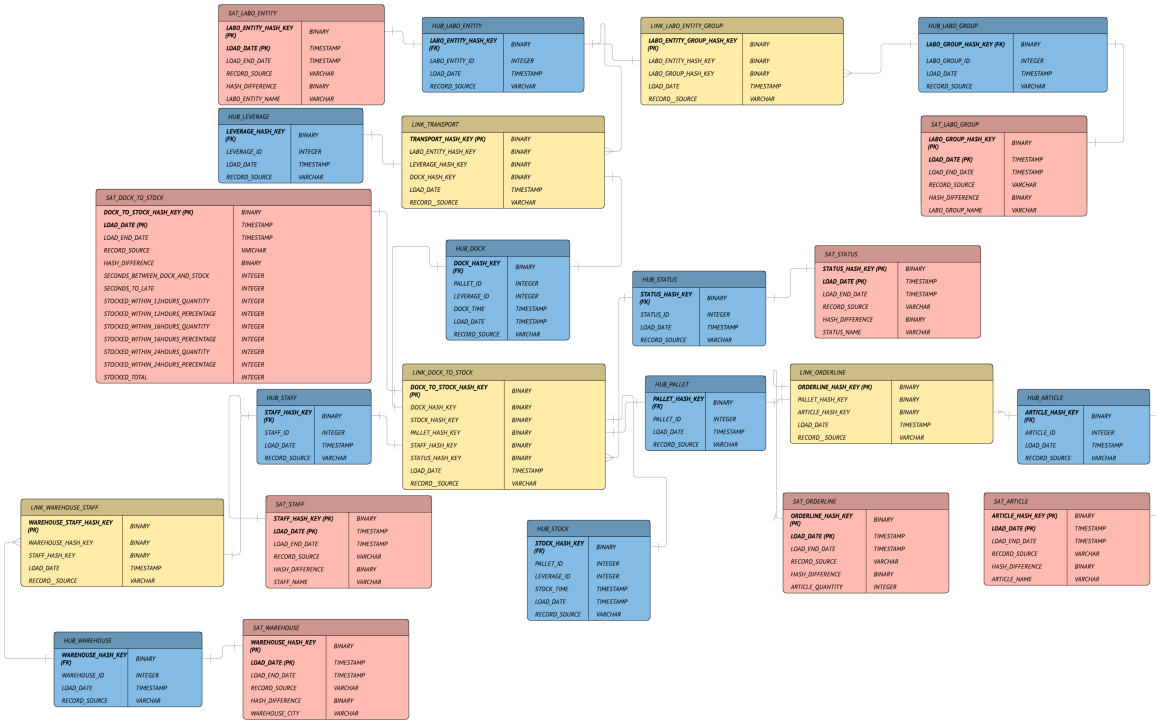
Figuur 3.3: Een remote connectie opgezet naar de host vanuit SAP HANA.

[1]Een bronsysteem kan meerdere databronnen bevatten (bijvoorbeeld zowel CSV-bestanden als een MySQL databank). Daarom is het noodzakelijk de juiste adapter te registreren bij SAP HANA.

3.6 Data Vault: data warehousing

In deze sectie wordt een data warehouse opgebouwd aan de hand van Data Vault. Elke laag van de architectuur (en de bijhorende ETL) komt aan bod. Op het einde van dit hoofdstuk moet er via een rapporteringsomgeving kunnen verbonden worden met een ontworpen data mart, gebaseerd op een data warehouse ontworpen met de Data Vault methodologie, om de benodigde gegevens te kunnen opvragen.

3.6.1 Overzicht datamodel



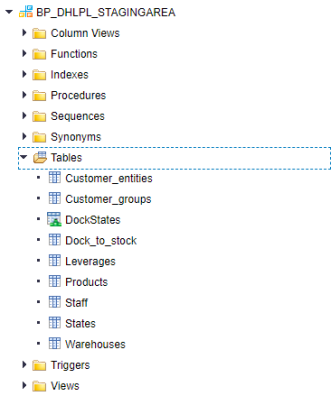
Figuur 3.4: Voorstelling van het Data Vault model (gemaakt via Lucidchart.com).

Het model in figuur 3.4 is opgebouwd uit drie soorten tabellen: de rode entiteiten worden voorgesteld als sattelites, de blauwe entiteiten als hubs en de gele entiteiten vormen de links tussen de verschillende hubs. In de SAT_DOCK_TO_STOCK worden de berekeningen opgeslagen die nodig zijn voor het berekenen van de KPI. Hash keys worden opgeslagen onder het type ”binary”.

3.6.2 Staging area

In de staging area worden alle gegevens in de originele vorm ingeladen. Dit betekent dat hierop nog geen manipulaties mogen gebeuren. In dit onderzoek wordt de data ingeladen via een virtuele tabel, afkomstig van een remote source (zie figuur 3.5).

Eens alle virtuele tabellen toegevoegd zijn in de staging area, dan is het modelleren van de



Figuur 3.5: Toevoegen van virtuele tabellen aan de staging area (SAP HANA).

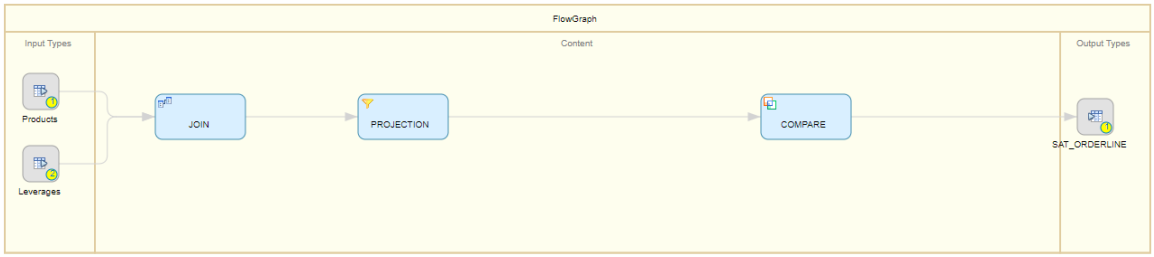
eerste laag afgewerkt en kan er overgegaan worden naar het modelleren van de raw Data Vault.

3.6.3 Opbouw raw Data Vault

In de raw Data Vault wordt de source data omgevormd naar de Data Vault methodologie. Hierbij wordt nog geen extra business logica en/of berekeningen toegevoegd. Concreet voor dit data model zullen de gegevens getransformeerd worden en doorgeladen worden naar alle tabellen, behalve de tabel SAT_DOCK_TO_STOCK (business logica). De data bij de tabel HUB_DOCK_TO_STOCK kan wel al ingeladen worden, aangezien deze geen business logica bevat (een hub of link bevat nooit business logica).

Inladen van data bij een sattelite

Het inladen & transformeren van de data bij alle sattelites binnen de raw Data Vault gebeurt gelijkaardig. In dit voorbeeld wordt het ETL-proces weergegeven van SAT_ORDERLINE en uitgelegd welke stappen ondernomen moeten worden om de data juist in te laden.



Figuur 3.6: Een voorbeeld van een ETL proces in SAP HANA bij een sattelite (SAP SDI).

In de eerste stap van dit proces worden verschillende bronnen (afkomstig uit de staging area) samengevoegd naar 1 dataset op basis van een gemeenschappelijk attribuut. Alle onnodige data wordt niet meegenomen naar de volgende stap. Deze stap kan ook overgeslagen worden indien de benodigde data afkomstig is uit één bron.

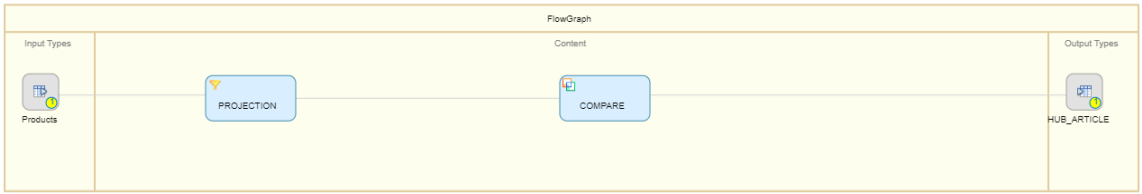
Vervolgens wordt de data getransformeerd naar het juiste formaat (naar het formaat van de destination tabel). Bij de tabel SAT_ORDERLINE worden volgende manipulaties uitgevoerd:

- **ORDERLINE_HASH_KEY (PK):** Een gehashte sleutel van volgende componenten: ”PRODUCT_ID” (tabel Products), ”PALLET_ID” (tabel Leverages) en de naam van het bronsysteem (in dit geval: ”DATAFILES”).
- **LOAD_DATE (PK):** Datum/tijdstip wanneer de record geëxtraheerd werd uit de bron.
- **LOAD_END_DATE:** Tijdstip tot wanneer de data actueel was, indien deze nog steeds actueel is, krijgt deze de waarde ”9999/12/31 23:59:59” (belangrijk voor de historiek).
- **RECORD_SOURCE:** De naam van het bronsysteem waarvan de data afkomstig is (”DATAFILES” in dit geval).
- **HASH_DIFFERENCE:** Alle data afkomstig in de entiteit die wordt opgenomen in het Data Vault model, wordt gehasht naar 1 sleutel. In dit geval wordt enkel ”ARTICLE_QUANTITY” versleuteld.
- **ARTICLE_QUANTITY:** Het attribuut ”PRODUCT_QUANTITY” wordt overgenomen van de brondata.

Nadat de transformaties gebeurd zijn, vergelijken we de nieuwe data met de data die in de destination table zit. Indien de data een nieuwere versies bevat, dan zal deze toegevoegd worden en de ”LOAD_END_DATE” van de oude entiteit gewijzigd worden naar het tijdstip van extractie van de nieuwe entiteit.

Inladen van data bij een hub

In elke hub worden de entiteiten bijgehouden die gebruikt zullen worden, en die het meest geschikt zijn voor de rapportering. Deze kunnen ook opgebouwd worden aan de hand van het principe van de "golden record", waarbij de data van verschillende bronnen samengevoegd worden tot één record, om zo een compleet en correct mogelijke record te verkrijgen. In dit onderzoek is dit niet het geval, aangezien er gewerkt wordt met één bron.



Figuur 3.7: Een voorbeeld van een ETL proces in SAP HANA bij een hub (SAP SDI).

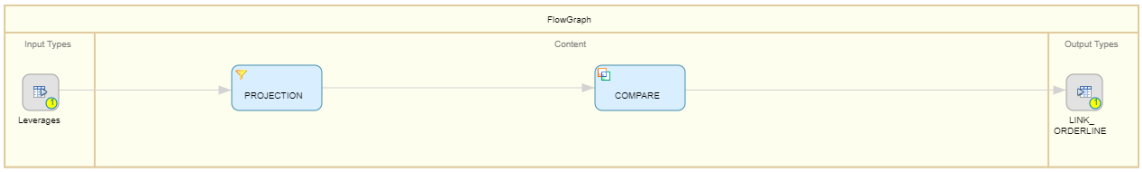
Bij de projectie worden volgend transformaties toegepast:

- **ARTICLE_HASH_KEY (PK):** Een gehashte sleutel van volgende componenten: ”PRODUCT_ID” (tabel Products) en de naam van het bronsysteem (in dit geval: ”DATAFILES”).
- **ARTICLE_ID:** De business key van de entiteit ”Article” wordt overgenomen van

de brondata.
- **LOAD_DATE:** Datum/tijdstip wanneer de record geëxtraheerd werd uit de bron.
- **RECORD_SOURCE:** De naam van het bronsysteem waarvan de data afkomstig is (”DATAFILES” in dit geval).

Inladen van data bij een link

Bij het inladen voor de data in de link-entiteiten worden dezelfde stappen doorlopen als bij het inladen van data bij hubs en sattelites. Links stellen de relaties voor tussen de verschillende entiteiten.



Figuur 3.8: Een voorbeeld van een ETL proces in SAP HANA bij een link (SAP SDI).

- **ORDERLINE_HASH_KEY (PK):** Een gehashte sleutel van volgende componenten: ”PRODUCT_ID” (tabel Leverages), ”PALLET_ID” (tabel Leverages) en de naam van het bronsysteem (in dit geval: ”DATAFILES”).
- **ARTICLE_HASH_KEY:** Een gehashte sleutel van volgende componenten: ”PRODUCT_ID” (tabel Leverages).
- **PALLET_HASH_KEY:** Een gehashte sleutel van volgende component: ”PALLET_ID” (tabel Leverages).
- **LOAD_DATE:** Datum/tijdstip wanneer de record geëxtraheerd werd uit de bron.
- **RECORD_SOURCE:** De naam van het bronsysteem waarvan de data afkomstig is (”DATAFILES” in dit geval).

3.6.4 Opbouw business vault

Aangezien de Business vault in dit data schema slechts 1 nieuwe entiteit bevat, vertelt de methodologie van Data Vault dat het niet nodig is om een nieuwe laag hiervoor te ontwerpen (besparen van opslagruimte). De business vault tabellen mogen dan toegevoegd worden aan de raw Data Vault indien dit de complexiteit niet aanzienlijk zou verhogen.

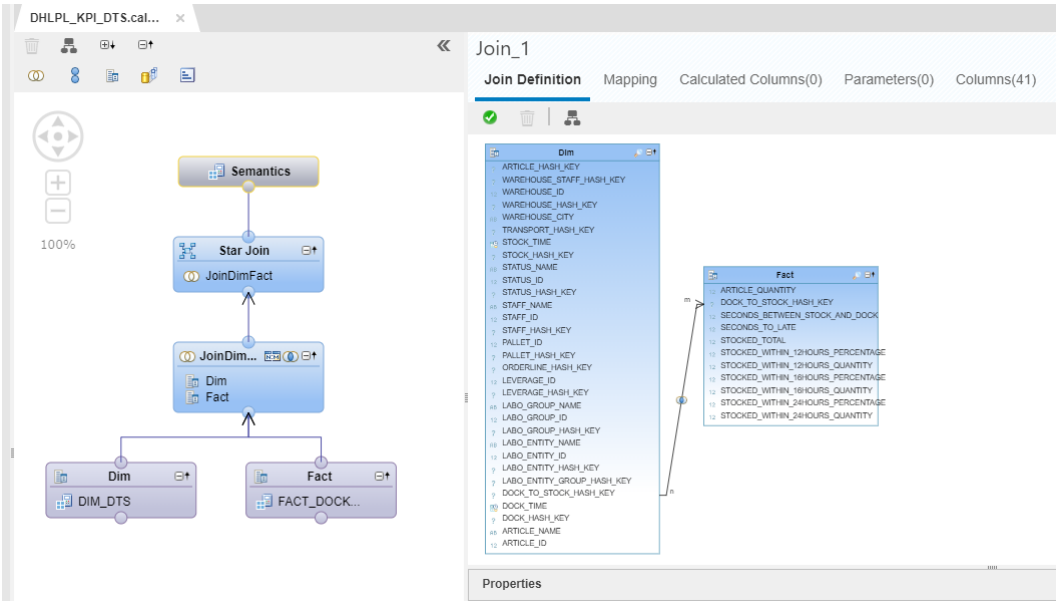
Bij het aanmaken van het ETL proces bij een sattelite binnen de Business vault, worden dezelfde stappen overlopen als bij het aanmaken van een sattelite binnen de raw Data Vault. Bij ”projection” wordt dan de benodigde business logica en de benodigde berekeningen toegevoegd.

3.6.5 Opbouw data mart

Wanneer alle data getransformeerd en ingeladen is in de Data Vault modellen, moeten deze verbonden worden met elkaar en wordt hiervoor een virtuele view aangemaakt waarop

verbonden kan worden vanuit een rapporteringsomgeving (zie figuur 3.9).

Een best practice die in dit onderzoek werd toegepast was het samenvoegen van alle hubs, links & sattelites in één dimensie. Zo wordt een beter overzicht behouden van de opgestelde calculation views en wordt de complexiteit verminderd. Deze dimensie werd dan verbonden met een fact table in een sterschema.



Figuur 3.9: Sterschema opgesteld in SAP voor Data Vault.

3.7 Dimensioneel model: data warehousing

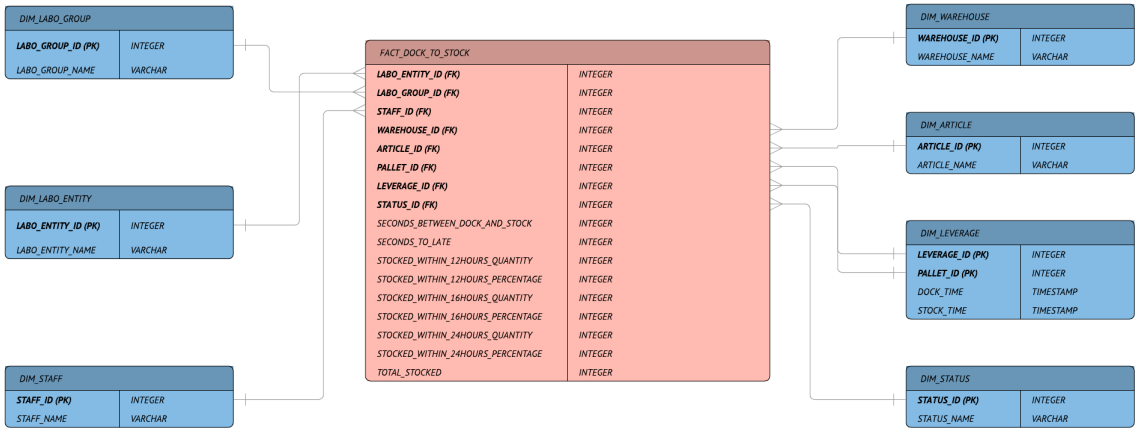
In dit hoofdstuk wordt een data warehouse opgebouwd aan de hand van het dimensioneel modelleren. Net zoals bij het Data Vault model, moet hier alles geconfigureerd worden zodat een verbinding mogelijk is vanuit een rapporteringsomgeving.

3.7.1 Overzicht datamodel

In figuur 3.10 worden de dimensions voorgesteld als de blauwe entiteiten. Deze bevatten de beschrijvende data die iets meer vertellen over de ”facts”. De business key wordt gebruikt als attribuut die de relatie legt naar de facts-table (die wordt voorgesteld in het rood). In dit model worden geen gegevens opgeslagen die meer vertellen over de oorsprong van de data, tevens wordt de historiek van de data niet bijgehouden.

3.7.2 Staging area

In de architectuur van Data Vault, is al reeds een staging area toegevoegd die alle informatie ongemanipuleerd bijhoudt. Voor dit onderdeel van het onderzoek zal de laag niet opnieuw



Figuur 3.10: Voorstelling van het dimensioneel model (gemaakt via Lucidchart.com).

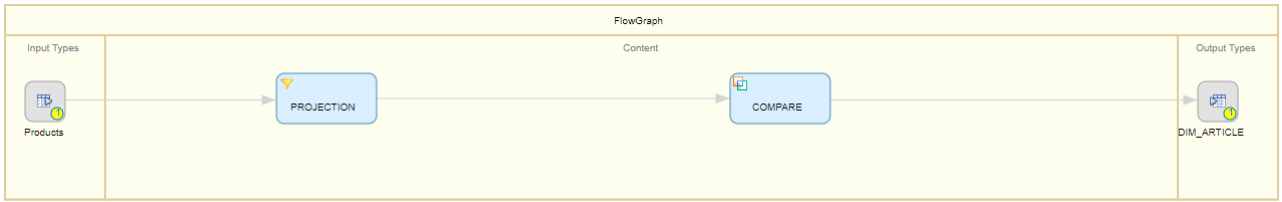
worden toegevoegd, maar zal er gebruik gemaakt worden van de eerder toegevoegde laag (zie sectie 3.6.2).

3.7.3 Opbouw data warehouselaag

In deze sectie wordt de data warehouselaag opgebouwd. In tegenstelling tot Data Vault wordt alle data in deze laag weggeschreven, inclusief de business logica die berekend moet worden. De benodigde data wordt opgehaald uit de staging area.

Dimensions

Bij het inladen van de dimensions bij het dimensioneel model, moeten er nooit berekeningen uitgevoerd worden, aangezien deze de beschrijvende data bevatten over de facts. Wel kan de data in dit proces gecleaned en gemanipuleerd worden om een betere structuur te krijgen.



Figuur 3.11: Voorstelling van het ETL-proces bij een dimension (SAP SDI).

In figuur 3.11 wordt het ETL-proces voorgesteld bij het inladen van de data in de tabel DIM_ARTICLE. Aangezien er enkel data moet ingeladen worden vanuit één enkele tabel, moet er geen JOIN gebeuren. Er kan dus onmiddellijk begonnen worden met het transformeren van de data in PROJECTION.

- **ARTICLE_ID (PK):** In dit attribuut wordt PRODUCT_ID overgenomen van de

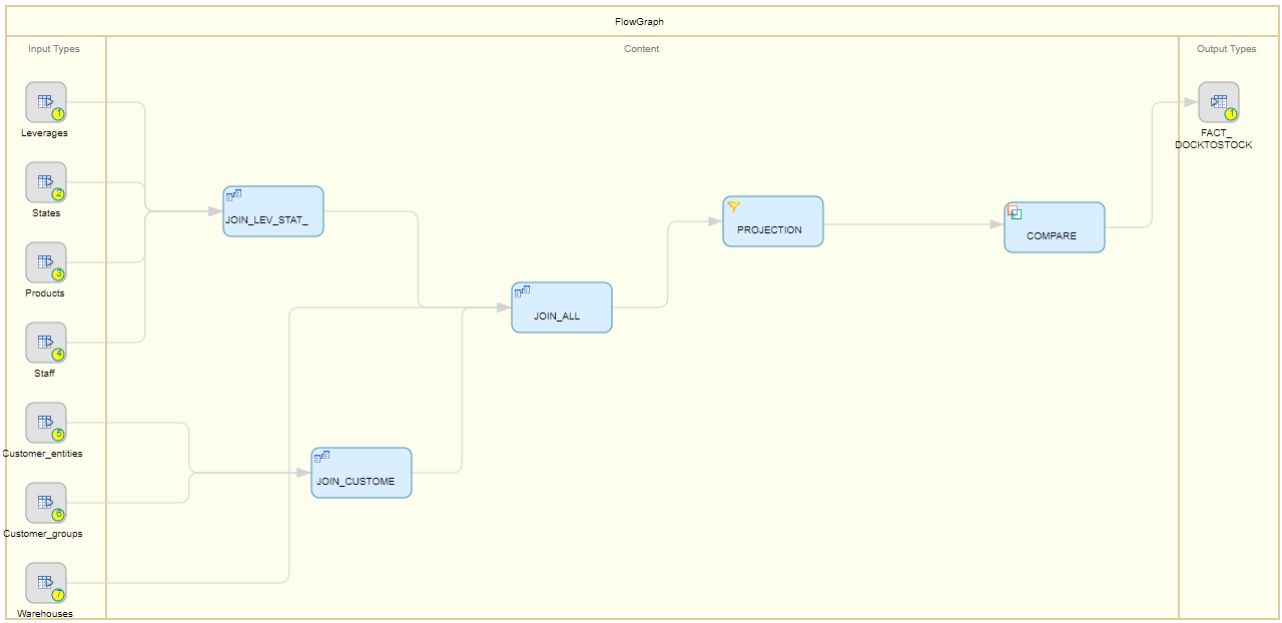
brondata.

- **ARTICLE_NAME:** In dit attribuut wordt PRODUCT_DESCRIPTION overgenomen van de brondata.

In de stap COMPARE wordt vergeleken of de nieuwe record al aanwezig is in de databank. Indien dit niet het geval is, wordt deze toegevoegd, anders wordt de record in de databank met dezelfde key overschreven. Bij het dimensioneel model wordt geen historiek bijgehouden van gegevens.

Facts

Bij een fact tabel, wordt de key van elke dimension bijgehouden. Alle sleutels samen vormen een composite key die dan geldt als primary key. Dit zorgt voor een zekere complexiteit bij het opbouwen van het ETL-proces. In dit proces worden ook de berekeningen uitgevoerd die nodig zijn voor het uitrekenen of de KPI wel/niet bereikt is.



Figuur 3.12: Voorstelling van het ETL-proces bij een fact (SAP SDI).

In de eerste stap van figuur 3.12 (JOIN_LEV_STAT_PROD_STAFF) wordt de data van de tabellen Leverages, States, Products en Staff samengevoegd op basis van de aanwezige attributen in de source tabel Leverages. Alleen de benodigde data wordt meegenomen naar de volgende stap. Bij JOIN_CUSTOMER wordt tussen de tabellen Customer_Entities en Customer_Groups een relatie gelegd. Om uiteindelijk alles samen te voegen tot één geheel, wordt er een finale join (JOIN_ALL) aangelegd. Hierbij wordt de data van JOIN_LEV_STAT_PROD_STAFF, JOIN_CUSTOMER en Warehouses gecombineerd die kan gebruikt worden in de volgende stappen.

Vervolgens wordt de benodigde data berekend. Volgende transformaties worden toegepast:

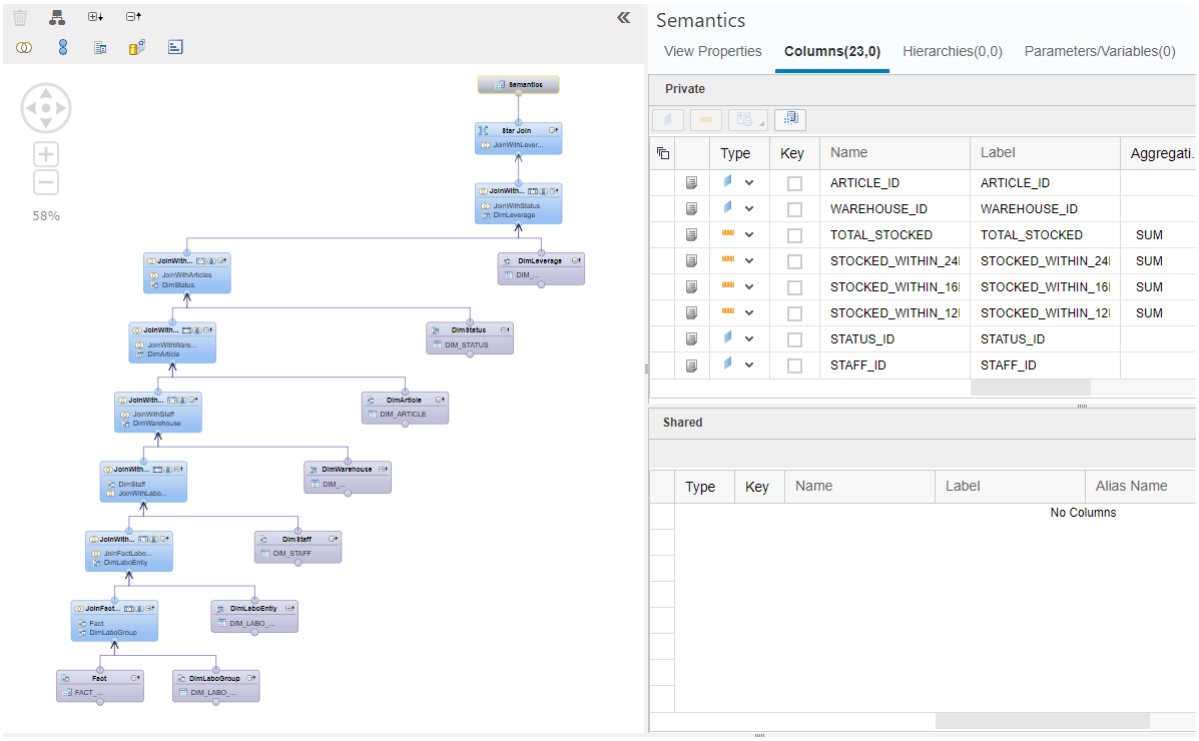
- **STAFF_ID (PK):** In dit attribuut wordt STAFF_ID overgenomen van de brondata.

- **ARTICLE_ID (PK):** In dit attribuut wordt PRODUCT_ID overgenomen van de brondata.
- **STATUS_ID (PK):** In dit attribuut wordt STATUS_ID overgenomen van de brondata.
- **WAREHOUSE_ID (PK):** In dit attribuut wordt WAREHOUSE_ID overgenomen van de brondata.
- **LABO_ENTITY_ID (PK):** In dit attribuut wordt LABO_ENTITY_ID overgenomen van de brondata.
- **LEVERAGE_ID (PK):** In dit attribuut wordt LEVERAGE_ID overgenomen van de brondata.
- **PALLET_ID (PK):** In dit attribuut wordt PALLET_ID overgenomen van de brondata.
- **LABO_GROUP_ID (PK):** In dit attribuut wordt LABO_GROUP_ID overgenomen van de brondata.
- **ARTICLE_QUANTITY :** In dit attribuut wordt PRODUCT_QUANTITY overgenomen van de brondata.
- **TOTAL_STOCKED :** In dit attribuut wordt het getal '1' opgeslagen. Zo kan er na de aggregatie een deling uitgevoerd worden om een bepaald percentage uit te rekenen. Tevens zorgt dit ook voor een duidelijke naam voor de business.
- **SECONDS_TO_LATE :** Indien het verschil tussen STOCK_TIJD en DOCK_TIJD kleiner is dan 86400 seconden (24u), dan is het resultaat 0. Indien dat verschil groter is dan 86400 seconden, dan wordt het verschil tussen STOCK_TIJD en DOCK_TIJD weergegeven.
- **SECONDS_BETWEEN_DOCK_AND_STOCK :** In dit attribuut wordt het verschil tussen STOCK_TIJD en DOCK_TIJD weergegeven.
- **STOCKED_WITHIN_12HOURS_QUANTITY :** Indien de pallet gestockeerd werd binnen de 12 uur tijd, dan is de waarde van dit attribuut 1, anders 0.
- **STOCKED_WITHIN_16HOURS_QUANTITY :** Indien de pallet gestockeerd werd binnen de 16 uur tijd, dan is de waarde van dit attribuut 1, anders 0.
- **STOCKED_WITHIN_24HOURS_QUANTITY :** Indien de pallet gestockeerd werd binnen de 24 uur tijd, dan is de waarde van dit attribuut 1, anders 0.

Nadat alle transformaties en berekeningen zijn toegepast, worden de toegekomen waardes vergeleken met de huidige waardes van de destination table. Er wordt vergeleken op basis van de composite key. Indien de waarde al in de database aanwezig is, wordt deze overschreven, anders wordt deze toegevoegd aan de dataset.

3.7.4 Opbouw data mart

Nadat alle tabellen in het dimensioneel model zijn aangevuld, kan er een data mart aangemaakt worden voor de opgestelde KPI. Dit gebeurt gelijkaardig zoals bij het Data Vault model. Er zal een sterschema aangemaakt worden waarbij alle dimensions verbonden worden (zie figuur 3.13) met de fact tabel. Hierop kan dan verbonden worden vanuit een rapporteringsomgeving.



Figuur 3.13: Sterschema opgesteld in SAP voor het dimensioneel model.

4. Vergelijkend Onderzoek

In dit deel van het onderzoek zal er van de twee eerder opgebouwde datamodellen een vergelijking gemaakt worden. Hierbij wordt voornamelijk gefocust op deze 5 pijlers:

- **Performantie:** is er een significant verschil in het uitvoeren van leesopdrachten?
- **Complexiteit:** zijn beide modellen makkelijk interpreteerbaar door IT & business?
- **Flexibiliteit:** hoe flexibel zijn beide modellen wanneer een business requirement gewijzigd wordt?
- **Schaalbaarheid:** hoe gaan beide modellen overweg met het inladen van enorme hoeveelheid data?
- **Audit:** is er metadata beschikbaar over de werkelijke data? Kunnen problemen makkelijk opgespoord worden?

Op basis van deze resultaten zal er kunnen opgemaakt worden of de keuze voor Data Vault wel de juiste keuze was voor dit project.

4.1 Performantie

In deze sectie wordt het vergelijkend onderzoek gestart door de performantie bij leesopdrachten te vergelijken tussen Data Vault en het dimensioneel model. Voor het verkrijgen van de resultaten wordt een procedure geschreven in SQL-code die de executietijd zal berekenen. Elke query zal 1000 keer uitgevoerd worden. Op basis van de verkregen resultaten kan er een correcte analyse opgemaakt worden. Deze queries worden uitgevoerd op een **SAP HANA databank**, door het gebruik te maken van deze technologie zal dit een invloed hebben op de snelheid van executie. Zowel voor Data Vault als voor het dimensioneel werden de leesopdrachten uitgevoerd met 2 verschillende hoeveelheden van

data. De 1000 leesoperaties werden uitgevoerd op 1200 en 100.000 records waarbij alle tabellen van ieder model opgevraagd worden. Al deze opgehaalde data zou dan kunnen gebruikt worden voor het opmaken van dashboards.

4.1.1 Procedure

```
CREATE PROCEDURE "TESTJORIK_BV"."RESULTS" LANGUAGE
    SQLSCRIPT AS BEGIN

--Declareren van variabele (teller)
DECLARE EXEC_ID int;

--Initialiseren van variabele
EXEC_ID = 1;

--Starten van loop (1000x queries uitvoeren)
WHILE :EXEC_ID < 1001 DO

--ID toevoegen in databank
INSERT INTO "TESTJORIK_BV"."UITVOERINGSTIJDEN" ("EXEC_ID")
    VALUES (:EXEC_ID);

--Tijdstip van start toevoegen in de databank voor het Data
    Vault model
UPDATE "TESTJORIK_BV"."UITVOERINGSTIJDEN" SET "
    START_TIME_DV" = CURRENT_TIMESTAMP
WHERE "EXEC_ID" = :EXEC_ID;

--Uitvoeren query leesopdracht op DataVault model
SELECT hubarticle."ARTICLE_ID", hubwarehouse."WAREHOUSE_ID"
    , hubstatus."STATUS_ID", hubstaff."STAFF_ID", hubpallet.
    "PALLET_ID", hubdelivery."DELIVERY_ID",
hubgroup."LABO_GROUP_ID", hubentity."LABO_ENTITY_ID",
    satgroup."LABO_GROUP_NAME", satentity."LABO_ENTITY_NAME"
    , satstaff."STAFF_NAME", satwarehouse."WAREHOUSE_CITY",
satarticle."ARTICLE_NAME", satstatus."STATUS_NAME", hubdock
    ."DOCK_TIME", hubstock."STOCK_TIME", satdts."
    STOCKED_TOTAL", satdts."STOCKED_WITHIN_12HOURS_QUANTITY"
    ,
satdts."STOCKED_WITHIN_16HOURS_QUANTITY", satdts."
    STOCKED_WITHIN_24HOURS_QUANTITY", satdts."
    SECONDS_TO_LATE", satdts."SECONDS_BETWEEN_STOCK_AND_DOCK
    ", satorderline."ARTICLE_QUANTITY"
FROM "BP_DHLPL_RAWDATAVAULT"."LINK_DOCK_TO_STOCK" as
    linkdts
INNER JOIN "BP_DHLPL_RAWDATAVAULT"."HUB_DOCK" as hubdock
```

```
ON linkdts."DOCK_HASH_KEY" = hubdock."DOCK_HASH_KEY"
INNER JOIN "BP_DHLPL_RAWDATAVAULT"."HUB_STOCK" as hubstock
ON linkdts."STOCK_HASH_KEY" = hubstock."STOCK_HASH_KEY"
INNER JOIN "BP_DHLPL_RAWDATAVAULT"."HUB_PALLET" as
    hubpallet
ON linkdts."PALLET_HASH_KEY" = hubpallet."PALLET_HASH_KEY"
INNER JOIN "BP_DHLPL_RAWDATAVAULT"."HUB_STATUS" as
    hubstatus
ON linkdts."STATUS_HASH_KEY" = hubstatus."STATUS_HASH_KEY"
INNER JOIN "BP_DHLPL_RAWDATAVAULT"."HUB_STAFF" as hubstaff
ON linkdts."STAFF_HASH_KEY" = hubstaff."STAFF_HASH_KEY"
INNER JOIN "BP_DHLPL_RAWDATAVAULT"."LINK_ORDERLINE" as
    linkorderline
ON hubpallet."PALLET_HASH_KEY" = linkorderline."
    PALLET_HASH_KEY"
INNER JOIN "BP_DHLPL_RAWDATAVAULT"."HUB_ARTICLE" as
    hubarticle
ON linkorderline."ARTICLE_HASH_KEY" = hubarticle."
    ARTICLE_HASH_KEY"
INNER JOIN "BP_DHLPL_RAWDATAVAULT"."LINK_WAREHOUSE_STAFF"
    as linkwarehousestaff
ON hubstaff."STAFF_HASH_KEY" = linkwarehousestaff."
    STAFF_HASH_KEY"
INNER JOIN "BP_DHLPL_RAWDATAVAULT"."HUB_WAREHOUSE" as
    hubwarehouse
ON linkwarehousestaff."WAREHOUSE_HASH_KEY" = hubwarehouse."
    WAREHOUSE_HASH_KEY"
INNER JOIN "BP_DHLPL_RAWDATAVAULT"."LINK_TRANSPORT" as
    linktransport
ON hubdock."DOCK_HASH_KEY" = linktransport."DOCK_HASH_KEY"
INNER JOIN "BP_DHLPL_RAWDATAVAULT"."HUB_LABO_ENTITY" as
    hubentity
ON linktransport."LABO_ENTITY_HASH_KEY" = hubentity."
    LABO_ENTITY_HASH_KEY"
INNER JOIN "BP_DHLPL_RAWDATAVAULT"."HUB_DELIVERY" as
    hubdelivery
ON linktransport."DELIVERY_HASH_KEY" = hubdelivery."
    DELIVERY_HASH_KEY"
INNER JOIN "BP_DHLPL_RAWDATAVAULT"."LINK_LABO_ENTITY_GROUP"
     as linklabo
ON hubentity."LABO_ENTITY_HASH_KEY" = linklabo."
    LABO_ENTITY_HASH_KEY"
INNER JOIN "BP_DHLPL_RAWDATAVAULT"."HUB_LABO_GROUP" as
    hubgroup
ON linklabo."LABO_GROUP_HASH_KEY" = hubgroup."
    LABO_GROUP_HASH_KEY"
```

```
INNER JOIN "BP_DHLPL_RAWDATAVAULT"."SAT_LABO_GROUP" as
    satgroup
ON satgroup."LABO_GROUP_HASH_KEY" = hubgroup."
    LABO_GROUP_HASH_KEY"
INNER JOIN "BP_DHLPL_RAWDATAVAULT"."SAT_LABO_ENTITY" as
    satentity
ON satentity."LABO_ENTITY_HASH_KEY" = hubentity."
    LABO_ENTITY_HASH_KEY"
INNER JOIN "BP_DHLPL_RAWDATAVAULT"."SAT_STATUS" as
    satstatus
ON hubstatus."STATUS_HASH_KEY" = satstatus."STATUS_HASH_KEY
    "
INNER JOIN "BP_DHLPL_RAWDATAVAULT"."SAT_STAFF" as satstaff
ON hubstaff."STAFF_HASH_KEY" = satstaff."STAFF_HASH_KEY"
INNER JOIN "BP_DHLPL_RAWDATAVAULT"."SAT_WAREHOUSE" as
    satwarehouse
ON hubwarehouse."WAREHOUSE_HASH_KEY" = satwarehouse."
    WAREHOUSE_HASH_KEY"
INNER JOIN "BP_DHLPL_RAWDATAVAULT"."SAT_ARTICLE" as
    satarticle
ON hubarticle."ARTICLE_HASH_KEY" = satarticle."
    ARTICLE_HASH_KEY"
INNER JOIN "BP_DHLPL_RAWDATAVAULT"."SAT_ORDERLINE" as
    satorderline
ON linkorderline."ORDERLINE_HASH_KEY" = satorderline."
    ORDERLINE_HASH_KEY"
INNER JOIN "BP_DHLPL_RAWDATAVAULT"."SAT_DOCK_TO_STOCK" as
    satdts
ON linkdts."DOCK_TO_STOCK_HASH_KEY" = satdts."
    DOCK_TO_STOCK_HASH_KEY";

--Tijdstip van einde toevoegen in de databank voor het Data
    Vault model
UPDATE "TESTJORIK_BV"."UITVOERINGSTIJDEN" SET "END_TIME_DV"
    = CURRENT_TIMESTAMP
WHERE "TESTJORIK_BV"."UITVOERINGSTIJDEN"."EXEC_ID" = :
    EXEC_ID;

--Tijdstip van start toevoegen in de databank voor het
    dimensioneel model
UPDATE "TESTJORIK_BV"."UITVOERINGSTIJDEN" SET "
    START_TIME_DM" = CURRENT_TIMESTAMP
WHERE "TESTJORIK_BV"."UITVOERINGSTIJDEN"."EXEC_ID" = :
    EXEC_ID;

--Uitvoeren query leesop^dracht op het dimensioneel model
```

```
SELECT fct.*, dimarticle."ARTICLE_NAME", dimentity."
    LABO_ENTITY_NAME", dimgroup."LABO_GROUP_NAME",
    dimdelivery."STOCK_TIME", dimdelivery."DOCK_TIME",
    dimstaff."STAFF_NAME", dimstatus."STATUS_NAME",
    dimwarehouse."WAREHOUSE_CITY"
FROM "BP_DHLPL_DATAWAREHOUSELAAG"."FACT_DOCKTOSTOCK" as fct
INNER JOIN "BP_DHLPL_DATAWAREHOUSELAAG"."DIM_ARTICLE" as
    dimarticle
ON fct."ARTICLE_ID" = dimarticle."ARTICLE_ID"
INNER JOIN "BP_DHLPL_DATAWAREHOUSELAAG"."DIM_LABO_ENTITY"
    as dimentity
ON fct."LABO_ENTITY_ID" = dimentity."LABO_ENTITY_ID"
INNER JOIN "BP_DHLPL_DATAWAREHOUSELAAG"."DIM_LABO_GROUP" as
     dimgroup
ON fct."LABO_GROUP_ID" = dimgroup."LABO_GROUP_ID"
INNER JOIN "BP_DHLPL_DATAWAREHOUSELAAG"."DIM_DELIVERY" as
    dimdelivery
ON fct."PALLET_ID" = dimdelivery."PALLET_ID"
INNER JOIN "BP_DHLPL_DATAWAREHOUSELAAG"."DIM_STAFF" as
    dimstaff
ON fct."STAFF_ID" = dimstaff."STAFF_ID"
INNER JOIN "BP_DHLPL_DATAWAREHOUSELAAG"."DIM_STATUS" as
    dimstatus
ON fct."STATUS_ID" = dimstatus."STATUS_ID"
INNER JOIN "BP_DHLPL_DATAWAREHOUSELAAG"."DIM_WAREHOUSE" as
    dimwarehouse
ON fct."WAREHOUSE_ID" = dimwarehouse."WAREHOUSE_ID";

--Tijdstip van einde toevoegen in de databank voor het
    dimensioneel model
UPDATE "TESTJORIK_BV"."UITVOERINGSTIJDEN" SET "END_TIME_DM"
     = CURRENT_TIMESTAMP
WHERE "TESTJORIK_BV"."UITVOERINGSTIJDEN"."EXEC_ID" = :
    EXEC_ID;

--Berekenen van executietijd voor het Data Vault model
UPDATE "TESTJORIK_BV"."UITVOERINGSTIJDEN" SET "DIFF_TIME_DV
    " = (NANO100_BETWEEN(START_TIME_DV,END_TIME_DV) / 10000)
WHERE "TESTJORIK_BV"."UITVOERINGSTIJDEN"."EXEC_ID" = :
    EXEC_ID;

--Berekenen van executietijd voor het dimensioneel model
UPDATE "TESTJORIK_BV"."UITVOERINGSTIJDEN" SET "DIFF_TIME_DM
    " = (NANO100_BETWEEN(START_TIME_DM,END_TIME_DM) / 10000)
WHERE "TESTJORIK_BV"."UITVOERINGSTIJDEN"."EXEC_ID" = :
    EXEC_ID;
```

```
--Waarde teller verhogen met 1
EXEC_ID = :EXEC_ID + 1;
END WHILE;
END;
```

4.1.2 Resultaten

Na het doorlopen van de vorige procedure, kan er gestart worden met het vergelijken van de gegenereerde data over de performantie bij leesopdrachten. In tabel 4.1 wordt een overzicht getoont van het werkgeheugen die SAP HANA toewees voor het uitvoeren van beide queries (door de query 1 keer uit te voeren).

	Data Vault	**Dimensioneel model**
1200 records	5,2 MByte(s)	2,5 MByte(s)
100.000 records	139,8 MByte(s)	4,6 MByte(s)

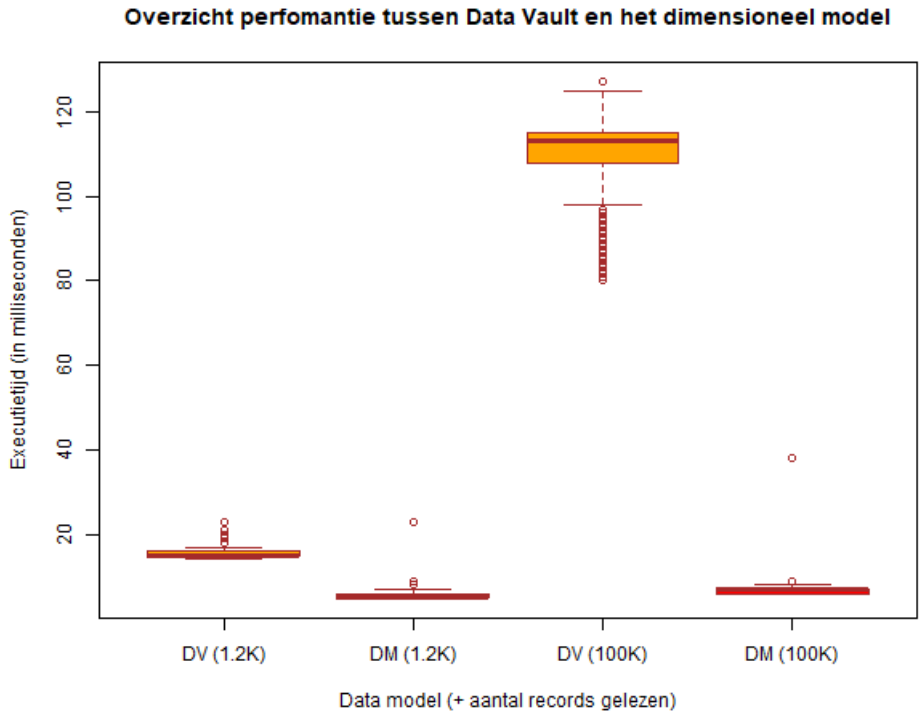
Tabel 4.1: Overzicht van het toegewezen werkgeheugen van beide select-statements (verkregen via het Execution plan in SAP HANA).

In tabel 4.2 wordt een overzicht getoont van het steekproefgemiddelde met de daarbijhorende standaardafwijking van de executietijd bij het uitvoeren van de leesopdracht bij beide modellen.

	Data Vault	**Dimensioneel model**
1200 records	15,29 ms ($\sigma = 0,772$)	5,466 ms ($\sigma = 0,783$)
100.000 records	108,086 ms ($\sigma = 11,375$)	6,783 ms ($\sigma = 1,122$)

Tabel 4.2: Overzicht van het steekproefgemiddelde bij Data Vault en het dimensioneel model.

In figuur 4.1 wordt een boxplot weergegeven met de snelheid bij leesopdrachten van beide data modellen.



Figuur 4.1: Boxplot van de uitvoeringstijden bij leesopdrachten van Data Vault en het dimensioneel model.

Uit deze boxplot kan afgeleid worden dat er een kleine spreiding is van de resultaten bij het dimensioneel model (op enkele uitschieters na). Opmerkelijk is dat de executietijd bij Data Vault bij het opvragen van 100.000 records opmerkelijk hoger ligt en er meer spreiding bij de data is in vergelijking met de executietijd bij het opvragen van 100.000 records bij het dimensioneel model.

4.1.3 Analyse van de resultaten

De resultaten die verkregen zijn zullen onderworpen worpen aan een statistische toets (t-toets aangezien de standaardafwijking van de populatie niet gekend is). Bij de resultaten tussen Data Vault en het dimensioneel model (bij een dataset van 1200 records) zijn volgende hypotheses bepaald:

$$H_0 = \mu_1 - \mu_2 = 0$$

$$H_1 = \mu_1 - \mu_2 > 0$$

waarbij μ_1 = gemiddelde executietijd bij Data Vault met 1.200 records & μ_2 = gemiddelde executietijd bij dimensioneel model met 1.200 records.

Er wordt een linkszijdige toets uitgevoerd omdat Data Vault veel meer joins bevat dan het dimensioneel model en dit zal een negatieve invloed hebben op het ophalen van de data. De vraag luidt of de resultaten significant zullen verschillen ten opzichte van het dimensioneel model.

Het betrouwbaarheidsinterval dat zal gehanteerd worden in deze toets is 95%. Dit betekent dat met 95% zekerheid kan gezegd worden dat een bepaalde parameter binnen het betrouwbaarheidsinterval zal liggen.

$$\alpha = 1 - 0.95$$
$$\alpha = 0.05$$

Het uitvoeren van de Welch Two Sample t-test in RStudio geeft volgend resultaat:

```
> t.test(resb$`DM (1.2K)`,resb$`DV (1.2K)`,alternative = "less")

        Welch Two Sample t-test

data:  resb$`DM (1.2K)` and resb$`DV (1.2K)`
t = -282.43, df = 1997.6, p-value < 2.2e-16
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
      -Inf -9.766758
sample estimates:
mean of x mean of y
    5.466    15.290
```

Figuur 4.2: Resultaat van de Welch Two Sample t-test in Rstudio voor de dataset met 1200 records tussen Data Vault en het dimensioneel model.

Aangezien de p-value uit de Welch Two Sample t-test (figuur 4.2) kleiner is dan α betekent dit dat de nullhypothese mag verworpen worden en dat de alternative hypothese mag aanvaard worden.

Wanneer er dezelfde methodologie wordt toegepast voor het bepalen ofdat er een significant verschil aanwezig tussen Data Vault en het dimensioneel model bij een dataset van 100.000 records, bekomen we volgend resultaat:

Na het uitvoeren van deze test (figuur 4.3) is de p-value hier ook kleiner dan α, ook hier betekent dit dat er een significant verschil aanwezig is tussen de performantie van beide data modellen.

Conclusie

Er kan geconcludeerd worden dat er toch wel degelijk een significant verschil zit tussen de gemiddelde executietijd van een leesopdracht bij Data Vault in vergelijking met het

```
> t.test(resb$`DM (100K)`,resb$`DV (100K)`,alternative = "less")

        Welch Two Sample t-test

data:  resb$`DM (100K)` and resb$`DV (100K)`
t = -280.27, df = 1018.4, p-value < 2.2e-16
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
      -Inf -100.7079
sample estimates:
mean of x mean of y
    6.783   108.086
```

Figuur 4.3: Resultaat van de Welch Two Sample t-test in Rstudio voor de dataset met 100.000 records tussen Data Vault en het dimensioneel model.

dimensioneel model. Niet alleen het data model, maar ook de hoeveelheid data aanwezig in de tabellen van een data model heeft een invloed op de snelheid van leesoperaties. Het dimensioneel model gaat hiermee duidelijk veel beter om dan het Data Vault model.

4.2 Complexiteit

Bij het modelleren van data kan complexiteit ontstaan zowel bij het interpreteren van het data model als bij het opmaken van het data model.

Wanneer de business of de IT een dimensioneel model moet interpreteren, zal dit gemakkelijker gaan in vergelijking met een Data Vault model. Bij een dimensioneel model zijn veel minder relaties, dat zorgt voor een overzichtelijker schema. Ook komt alles samen in één centraal punt (fact), waaraan alles gelinkt is. Bij het interpreteren van een Data Vault model, moet die persoon al beschikken over kennis over Data Vault. Hij/zij moet weten wat het doel is van de verschillende soorten tabellen (hubs, links & sattelites). Ook zullen de vele verschillende entiteiten samen het overzicht belemmeren.

De transactionele gegevens staan bij het dimensioneel model altijd gecentraliseerd op één plaats. Zo is het zeer eenvoudig om te weten te komen welke transactionele gegevens (en business logica) er gebruikt wordt. Dit is niet altijd het geval bij Data Vault. In dit onderzoek staan die gegevens toevallig wel samen op één plaats, maar in de praktijk zal dit niet altijd het geval zijn. Dan worden de transactionele gegevens bij een Data Vault model samengevoegd in een data mart.

Als een datamodel ontworpen wordt aan de hand van het dimensioneel modelleren, kan er snel en gemakkelijk tot een resultaat bekomen worden in vergelijking met Data Vault. Bij het ontwerpen van een dimensioneel model wordt er gestart vanuit de dimensions, die leiden naar één centraal punt, de fact. Wanneer een Data Vault model dient ontworpen te worden, kan dit op veel verschillende manieren gebeuren. Er wordt gestart vanuit hubs, die de pilaren vormen voor het model, daarna worden deze verbonden met elkaar aan de hand van links. Finaal wordt beslist welke sattelites zullen aangemaakt worden en aan welke hubs/links deze gekoppeld zullen worden.

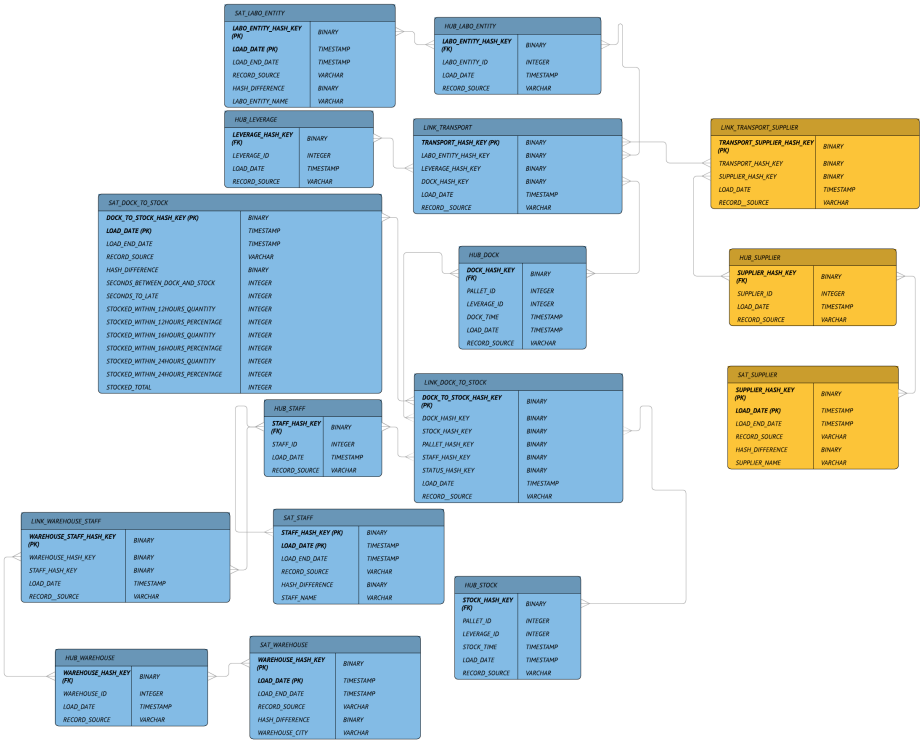
Wanneer historisatie van data een requirement is, dan heeft Data Vault een streepje voor op het dimensioneel modelleren. Bij Data Vault zit historisatie al in het model geïntegreerd,

bij het dimensioneel modelleren niet. Er kan geopteerd worden voor historisatie bij het dimensioneel modelleren door gebruik te maken van slow changing dimensions, maar dit verhoogt wel aanzienlijk de complexiteit van het model.

Op vlak van complexiteit, kan er geconcludeerd worden dat Data Vault een hogere complexiteit bevat in vergelijking met het dimensioneel model, zowel op het vlak van interpreteren van het model als het opstellen van een nieuw model.

4.3 Flexibiliteit

Bij het toevoegen van nieuwe requirements, moet er bij Data Vault nieuwe hub(s) ontworpen en geïmplementeerd worden samen met de daarbij horende links/sattelites. Indien daarbij nieuwe businesslogica wordt bijgevoegd, kan deze in de nieuwe sattelites geplaatst worden.



Figuur 4.4: Het toevoegen van een nieuwe entiteit bij Data Vault (gemaakt via Lucidchart.com).

In figuur 4.4 wordt een nieuwe business requirement toegevoegd bij een bestaand Data Vault model. Hierbij wordt het bestaand model niet aangepast, maar er wordt wel een uitbreiding uitgevoerd. Er wordt een nieuwe link, hub & sattelite toegevoegd (in het geel).

Deze manier van werken brengt een heel groot voordeel met zich mee op vlak van project management: via deze manier kan er gewerkt worden via de agile-manier, aangezien een toevoeging geen impact mag hebben op het bestaand data model.

Nieuwe business requirements toevoegen in het dimensioneel model is niet zo vanzelfsprekend. Een dimension ontwerpen en implementeren is eenvoudig, maar die nieuwe dimension zal moeten in relatie gezet worden met de centrale fact tabel. Dit betekent dat er een nieuwe key zal moeten toegevoegd worden in de fact tabel, en dat de huidige composite key dus zal moeten aangepast worden. De nieuwe toegevoegde key zal geen lege waarde mogen zijn.

Figuur 4.5: Het toevoegen van een nieuwe dimension bij een dimensioneel model (gemaakt via Lucidchart.com).

In figuur 4.5 wordt een nieuwe dimension (in het geel) toegevoegd aan het dimensioneel model. Het toevoegen van de nieuwe dimension heeft geen effect op de bestaande dimensions, maar wel in de fact tabel. In de fact tabel zal er een kolom moeten toegevoegd worden die de overeenkomstige keys bevat met de huidige data. Dit betekent dat het toevoegen van een nieuwe dimension wel degelijk een invloed heeft op het bestaande model en dat hier moeilijk via de agile methodologie kan gewerkt worden.

De structuur van een dimensioneel model ligt vast. Er is altijd een fact tabel die alle dimensions met elkaar verbindt. Het biedt geen opportuniteit om het schema te ontwerpen op basis van een business proces. Bij Data Vault is dit echter wel mogelijk. De flexibiliteit in de opbouw van een Data Vault is enorm hoog. Zo kan het model ontworpen worden op basis van een proces.

In een Data Vault model worden alleen maar many-to-many relaties gelegd door gebruik te maken van links. Dit is niet het geval bij het dimensioneel model. Door te werken met many-to-many relaties, verhoogt dit de flexibiliteit. Het nadeel door met many-to-many relaties te werken is dat het opvragen data langzamer zal verlopen.

Het toevoegen van nieuwe bron gebeurt doorgaans ook eenvoudig bij Data Vault. Hiervoor moeten enkel nieuwe sattelites aangemaakt worden voor de benodigde data. Deze sattelites worden dan verbonden met de bijhoorende hubs. Opnieuw wordt het bestaande model niet aangepast. Bij het dimensioneel model moet hiervoor gebruik gemaakt worden van slow changing dimensions. Als dit principe niet van in het begin opgenomen is in het model, dan zal er heel wat moeite moeten gestopt worden in het toevoegen van een nieuwe databron (extra kolommen aanmaken, bestaande ETL aanpassen, ...).

De conclusie bij de vergelijking van Data Vault en het dimensioneel op basis van flexibiliteit is zeer duidelijk. De grote winnaar bij flexibiliteit is Data Vault, dat op alle vlakken van dit aspect duidelijk de betere keuze is.

4.4 Schaalbaarheid

Data warehouses bevatten doorgaans enorme hoeveelheden data. Bij de vergelijking tussen Data Vault en het dimensioneel model is het belangrijk om de schaalbaarheid van beide modellen te bestuderen.

Het inladen van data in data warehouses kan bij enorme hoeveelheden data enorm veel tijd in beslag nemen. Daarom is het een enorm voordeel indien er kan gebruik gemaakt worden van parallel inladen, wat het geval is bij Data Vault en niet bij het dimensioneel modelleren.

Bij het inladen van data in het Data Vault model zijn alle tabellen onafhankelijk, dit wil zeggen dat alle tabellen tegelijk kunnen worden ingeladen. Bij het dimensioneel model moeten eerst de dimensions ingeladen zijn vooraleer de fact tabel kan ingeladen/aangevuld worden.

In deze sectie zullen 1000 keer 1 record ingeladen worden in alle tabellen van beide datamodellen. Hierbij worden de uitvoeringstijden van beide data modellen bijgehouden en vervolgens wordt een analyse opgemaakt op basis van de resultaten.

4.4.1 Procedure

```
ALTER PROCEDURE "TESTJORIK_BV"."ADDDATA" LANGUAGE SQLSCRIPT
    AS BEGIN
DECLARE START_TIME_DV timestamp;
DECLARE END_TIME_DV timestamp;
DECLARE TIME_DIFF_DV int;
DECLARE START_TIME_DM timestamp;
DECLARE END_TIME_DM timestamp;
DECLARE TIME_DIFF_DM int;
DECLARE EXEC_ID int;
DECLARE AANTAL int;

EXEC_ID = 1002;
DELETE FROM "TESTJORIK_BV"."UITVOERINGSTIJDEN";
WHILE :EXEC_ID < 2002 DO
INSERT INTO "TESTJORIK_BV"."UITVOERINGSTIJDEN" ("EXEC_ID")
   VALUES (:EXEC_ID);

UPDATE "TESTJORIK_BV"."UITVOERINGSTIJDEN" SET "
   START_TIME_DV" = CURRENT_TIMESTAMP
```

```
WHERE "EXEC_ID" = :EXEC_ID;

--add
BEGIN PARALLEL EXECUTION

UPSERT "BP_DHLPL_RAWDATAVAULT"."LINK_DOCK_TO_STOCK" VALUES
    (HASH_SHA256(TO_BINARY(:EXEC_ID)),CURRENT_TIMESTAMP,'
    dummy','1B3E2CAC1E858C5C078BE89E8DB166CA019A␣2
    C281C1971925C6AC7CFFFCE1EBC', '92F061FC6C9BC77EBAB163C6␣
    3118347701E3825928FF6A8FE75736DBCB0A4CB1',
'27BADC983DF1780B60C2B3FA␣9
    D3A19A00E46AAC798451F0FEBDCA52920FAADDF','7902699
    BE42C8A8E46FBBB45017␣26517
    E86B22C56A189F7625A6DA49081B2451', '4
    E07408562BEDB8B60CE05C1DECF␣
    E3AD16B72230967DE01F640B7E4729B49FCE');
UPSERT "BP_DHLPL_RAWDATAVAULT"."SAT_DOCK_TO_STOCK" VALUES (
    HASH_SHA256(TO_BINARY(:EXEC_ID)),CURRENT_TIMESTAMP,
    CURRENT_TIMESTAMP,'dummy','1',1,1,1,1,1,1,1,1,1);
UPSERT "BP_DHLPL_RAWDATAVAULT"."SAT_LABO_ENTITY" VALUES (
    HASH_SHA256(TO_BINARY(:EXEC_ID)),CURRENT_TIMESTAMP,
    CURRENT_TIMESTAMP,'dummy','1','naam');
UPSERT "BP_DHLPL_RAWDATAVAULT"."SAT_LABO_GROUP" VALUES (
    HASH_SHA256(TO_BINARY(:EXEC_ID)),CURRENT_TIMESTAMP,
    CURRENT_TIMESTAMP,'dummy','1','naam');
UPSERT "BP_DHLPL_RAWDATAVAULT"."SAT_STATUS" VALUES (
    HASH_SHA256(TO_BINARY(:EXEC_ID)),CURRENT_TIMESTAMP,
    CURRENT_TIMESTAMP,'dummy','1','naam');
UPSERT "BP_DHLPL_RAWDATAVAULT"."SAT_STAFF" VALUES (
    HASH_SHA256(TO_BINARY(:EXEC_ID)),CURRENT_TIMESTAMP,
    CURRENT_TIMESTAMP,'dummy','1','naam');
UPSERT "BP_DHLPL_RAWDATAVAULT"."SAT_WAREHOUSE" VALUES (
    HASH_SHA256(TO_BINARY(:EXEC_ID)),CURRENT_TIMESTAMP,
    CURRENT_TIMESTAMP,'dummy','1','naam');
UPSERT "BP_DHLPL_RAWDATAVAULT"."SAT_ORDERLINE" VALUES (
    HASH_SHA256(TO_BINARY(:EXEC_ID)),CURRENT_TIMESTAMP,
    CURRENT_TIMESTAMP,'dummy','1',1);
UPSERT "BP_DHLPL_RAWDATAVAULT"."SAT_ARTICLE" VALUES (
    HASH_SHA256(TO_BINARY(:EXEC_ID)),CURRENT_TIMESTAMP,
    CURRENT_TIMESTAMP,'dummy','1','naam');
UPSERT "BP_DHLPL_RAWDATAVAULT"."HUB_LABO_ENTITY" VALUES (
    HASH_SHA256(TO_BINARY(:EXEC_ID)),:EXEC_ID,
    CURRENT_TIMESTAMP,'dummy');
UPSERT "BP_DHLPL_RAWDATAVAULT"."HUB_LABO_GROUP" VALUES (
    HASH_SHA256(TO_BINARY(:EXEC_ID)),:EXEC_ID,
    CURRENT_TIMESTAMP,'dummy');
```

```
UPSERT "BP_DHLPL_RAWDATAVAULT"."HUB_STATUS" VALUES (
    HASH_SHA256(TO_BINARY(:EXEC_ID)),:EXEC_ID,
    CURRENT_TIMESTAMP,'dummy');
UPSERT "BP_DHLPL_RAWDATAVAULT"."HUB_STAFF" VALUES (
    HASH_SHA256(TO_BINARY(:EXEC_ID)),:EXEC_ID,
    CURRENT_TIMESTAMP,'dummy');
UPSERT "BP_DHLPL_RAWDATAVAULT"."HUB_WAREHOUSE" VALUES (
    HASH_SHA256(TO_BINARY(:EXEC_ID)),:EXEC_ID,
    CURRENT_TIMESTAMP,'dummy');
UPSERT "BP_DHLPL_RAWDATAVAULT"."HUB_ARTICLE" VALUES (
    HASH_SHA256(TO_BINARY(:EXEC_ID)),:EXEC_ID,
    CURRENT_TIMESTAMP,'dummy');
UPSERT "BP_DHLPL_RAWDATAVAULT"."HUB_PALLET" VALUES (
    HASH_SHA256(TO_BINARY(:EXEC_ID)),:EXEC_ID,
    CURRENT_TIMESTAMP,'dummy');
UPSERT "BP_DHLPL_RAWDATAVAULT"."HUB_DELIVERY" VALUES (
    HASH_SHA256(TO_BINARY(:EXEC_ID)),:EXEC_ID,
    CURRENT_TIMESTAMP,'dummy');
UPSERT "BP_DHLPL_RAWDATAVAULT"."HUB_DOCK" VALUES (
    HASH_SHA256(TO_BINARY(:EXEC_ID)),:EXEC_ID,:EXEC_ID,
    CURRENT_TIMESTAMP,CURRENT_TIMESTAMP,'dummy');
UPSERT "BP_DHLPL_RAWDATAVAULT"."HUB_STOCK" VALUES (
    HASH_SHA256(TO_BINARY(:EXEC_ID)),:EXEC_ID,:EXEC_ID,
    CURRENT_TIMESTAMP,CURRENT_TIMESTAMP,'dummy');
UPSERT "BP_DHLPL_RAWDATAVAULT"."LINK_ORDERLINE" VALUES (
    HASH_SHA256(TO_BINARY(:EXEC_ID)),CURRENT_TIMESTAMP,'
    dummy','1','1');
UPSERT "BP_DHLPL_RAWDATAVAULT"."LINK_LABO_ENTITY_GROUP"
    VALUES (HASH_SHA256(TO_BINARY(:EXEC_ID)),
    CURRENT_TIMESTAMP,'dummy','1','1');
UPSERT "BP_DHLPL_RAWDATAVAULT"."LINK_TRANSPORT" VALUES (
    HASH_SHA256(TO_BINARY(:EXEC_ID)),CURRENT_TIMESTAMP,'
    dummy','1','1','1');
UPSERT "BP_DHLPL_RAWDATAVAULT"."LINK_WAREHOUSE_STAFF"
    VALUES (HASH_SHA256(TO_BINARY(:EXEC_ID)),
    CURRENT_TIMESTAMP,'dummy','1','1');
END;

UPDATE "TESTJORIK_BV"."UITVOERINGSTIJDEN" SET "END_TIME_DV"
     = CURRENT_TIMESTAMP
WHERE "TESTJORIK_BV"."UITVOERINGSTIJDEN"."EXEC_ID" = :
    EXEC_ID;
UPDATE "TESTJORIK_BV"."UITVOERINGSTIJDEN" SET "
    START_TIME_DM" = CURRENT_TIMESTAMP
WHERE "TESTJORIK_BV"."UITVOERINGSTIJDEN"."EXEC_ID" = :
    EXEC_ID;
```

```
--dimmodel
BEGIN PARALLEL EXECUTION
UPSERT "BP_DHLPL_DATAWAREHOUSELAAG"."DIM_LABO_GROUP" VALUES
    (:EXEC_ID,'x');
UPSERT "BP_DHLPL_DATAWAREHOUSELAAG"."DIM_LABO_ENTITY"
  VALUES (:EXEC_ID,'x');
UPSERT "BP_DHLPL_DATAWAREHOUSELAAG"."DIM_STAFF" VALUES (:
  EXEC_ID,'x');
UPSERT "BP_DHLPL_DATAWAREHOUSELAAG"."DIM_WAREHOUSE" VALUES
  (:EXEC_ID,'x');
UPSERT "BP_DHLPL_DATAWAREHOUSELAAG"."DIM_ARTICLE" VALUES (:
  EXEC_ID,'x');
UPSERT "BP_DHLPL_DATAWAREHOUSELAAG"."DIM_STATUS" VALUES (:
  EXEC_ID,'x');
UPSERT "BP_DHLPL_DATAWAREHOUSELAAG"."DIM_DELIVERY" VALUES
  (:EXEC_ID,:EXEC_ID,CURRENT_TIMESTAMP,CURRENT_TIMESTAMP);
END;
UPSERT "BP_DHLPL_DATAWAREHOUSELAAG"."FACT_DOCKTOSTOCK"
  VALUES (1,1,1,1,1,1,:EXEC_ID,1,1,1,1,1,1,1,1);

UPDATE "TESTJORIK_BV"."UITVOERINGSTIJDEN" SET "END_TIME_DM"
    = CURRENT_TIMESTAMP
WHERE "TESTJORIK_BV"."UITVOERINGSTIJDEN"."EXEC_ID" = :
  EXEC_ID;

UPDATE "TESTJORIK_BV"."UITVOERINGSTIJDEN" SET "DIFF_TIME_DV
  " = (NANO100_BETWEEN(START_TIME_DV,END_TIME_DV) / 10000)
WHERE "TESTJORIK_BV"."UITVOERINGSTIJDEN"."EXEC_ID" = :
  EXEC_ID;

UPDATE "TESTJORIK_BV"."UITVOERINGSTIJDEN" SET "DIFF_TIME_DM
  " = (NANO100_BETWEEN(START_TIME_DM,END_TIME_DM) / 10000)
WHERE "TESTJORIK_BV"."UITVOERINGSTIJDEN"."EXEC_ID" = :
  EXEC_ID;
EXEC_ID = :EXEC_ID + 1;
END WHILE;
END;
```

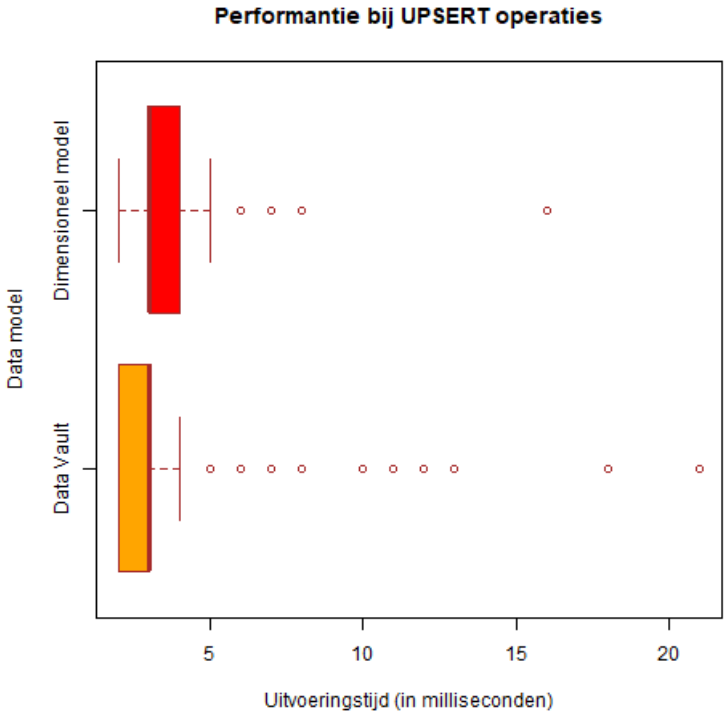
4.4.2 Resultaten

In tabel 4.3 wordt een overzicht weergegeven van de steekproefgemiddelden met de daarbijhorende standaardafwijking van de executietijd bij het uitvoeren van de UPSERT-operatie bij beide modellen.

	Data Vault	Dimensioneel model
1 record / tabel	3.109 ms ($\sigma = 1.438$)	3.411 ms ($\sigma = 0.781$)

Tabel 4.3: Overzicht van het steekproefgemiddelde bij Data Vault en het dimensioneel model bij het wegschrijven van data naar elke tabel.

Het verschil tussen de gemiddelde executietijden voor het invoegen van data is niet erg groot. Deze waarden zullen moeten onderworpen worden aan een statistische toets om te bepalen of er een significant verschil is.



Figuur 4.6: Boxplot van de uitvoeringstijden bij UPSERT-operaties van Data Vault en het dimensioneel model.

In figuur 4.6 wordt een boxplot weergegeven. Op deze figuur merken we op dat er een kleine spreiding is van de data bij beide data modellen, op enkele uitschieters na. De gemiddelde executietijd ligt dicht bij elkaar.

4.4.3 Analyse van de resultaten

Ook in deze subsectie zullen de resultaten die verkregen zijn onderworpen worpen aan een statistische toets (t-toets aangezien de standaardafwijking van de populatie niet gekend is). Volgende hypothesen worden opgemaakt:

$$H_0 = \mu_1 - \mu_2 = 0$$

$$H_1 = \mu_1 - \mu_2 \neq 0$$

waarbij μ_1 = gemiddelde executietijd bij Data Vault voor het toevoegen van een record bij elke tabel van het model & μ_2 = gemiddelde executietijd bij dimensioneel model voor het toevoegen van een record bij elke tabel van het model.

Het betrouwbaarheidsinterval dat zal gehanteerd worden in deze toets is ook 95%.

$$\alpha = 1 - 0.95 = 0.05$$

Het uitvoeren van de Welch Two Sample t-test in RStudio geeft volgend resultaat:

```
> t.test(data$`Data Vault`,data$`Dimensioneel model`)

        Welch Two Sample t-test

data:  data$`Data Vault` and data$`Dimensioneel model`
t = -5.8338, df = 1541.4, p-value = 6.592e-09
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.4035422 -0.2004578
sample estimates:
mean of x mean of y
    3.109     3.411
```

Figuur 4.7: Resultaat van de Welch Two Sample t-test in Rstudio voor het verschil in uitvoeringstijden bij een UPSERT-operatie.

Aangezien de p-value uit figuur 4.7 kleiner is dan α betekent dit dat ook deze nullhypothese mag verworpen worden en dat de alternative hypothese mag aanvaard worden. Er is dus een significant verschil aanwezig tussen beide data modellen bij het inladen van data.

4.4.4 Conclusie

Bij Data Vault kan de data significant sneller in het model geladen worden dan bij het dimensioneel model. Dit heeft te maken met het feit dat er bij het dimensioneel model eerst de dimensions moeten ingeladen zijn vooraleer de fact tabel kan ingeladen worden. Data Vault maakt gebruik van hash keys, hierdoor kan er sneller gecontroleerd worden of een bepaalde record al aanwezig is.

4.5 Audit

Het bijhouden van historisatie en extra informatie over gegevens is voor sommige organisaties van groot belang. Denk maar bijvoorbeeld aan de banksector, indien er iets fout is gelopen met een bepaalde transactie of indien men wil te weten komen van waar bepaalde data opgehaald wordt, moet er hiervoor extra informatie aangeleverd kunnen worden. Historisatie wordt voornamelijk gebruikt bij gevoelige data. Andere voorbeelden van sectoren zijn: medische sector, overheden, verzekeringen, ...

In de Data Vault methodologie zit historisatie verwerkt. Er wordt extra informatie bijgehouden over waar de data exact vandaan komt (RECORD_SOURCE), en wanneer de extractie van die data uit de bron gebeurd is (LOAD_DATE). De reden waarom het tijdstip van extractie wordt bijgehouden en niet het tijdstip van het inladen van de gegevens in de data warehouse is omdat het ETL proces zeer lang kan duren en in de tussentijd de data in het bronsysteem al gewijzigd kan zijn. Via de Data Vault methodologie worden er in principe geen rijen gewijzigd (enkel de kolom LOAD_END_DATE), en worden nieuwere versies van data gewoon toegevoegd in de databank.

Bij het dimensioneel model wordt standaard noch de historisatie van data, noch extra informatie over de aangeleverde data niet bijgehouden. Voor historisatie bij het dimensioneel model kan gebruik gemaakt worden van slow changing dimensions. Indien gewenst kan er natuurlijk ook altijd geopteerd worden om deze extra informatie op te nemen door enkele kolommen toe te voegen.

Er kan geconcludeerd worden dat Data Vault standaard historisatie toepast in hun model en dat er extra informatie (audit data) opgeslaan wordt over de data. Bij het dimensioneel model worden historisatie en het bijhouden van audit data niet opgenomen, maar het kan wel geïmplementeerd worden indien gewenst.

4.6 Overzicht

In tabel 4.4 wordt een overzicht weergegeven van de verschillen op basis van de 5 pijlers.

	Data Vault	**Dimensioneel model**
Performantie	Data opvragen gebeurt significant langzamer door de vele joins in het model.	Data lezen gebeurt vlot door een beperkt aantal joins.
Complexiteit	Hoge complexiteit, kennis nodig rond Data Vault.	Lage complexiteit, overzichtelijk.
Flexibiliteit	Zeer hoge flexibiliteit, mogelijk om via de agile methode te werken.	Zeer lage flexibiliteit, vaste structuur.
Schaalbaarheid	Goede schaalbaarheid door parallel inladen van de data.	Data inladen kan significant langer duren doordat de fact tabel afhankelijk is van de dimensions en er niet parallel kan ingeladen worden.
Audit	Extra informatie aanwezig en de historiek van de data wordt bijgehouden.	Extra informatie niet aanwezig, indien gewenst kan dit wel toegevoegd worden.

Tabel 4.4: Overzicht van het vergelijkend onderzoek.

5. Conclusie

Voor DHL Pharma Logistics werd gekozen om een data warehouse te ontwerpen aan de hand van de Data Vault methodologie. Was dit de juiste keuze? Was het dimensioneel model in hun project geen betere keuze? Wat wijst deze vergelijkende studie uit?

Het onderzoek wijst uit dat het dimensioneel model een betere keuze was voor het modelleren van de data voor DHL Pharma Logistics. Dit is bovendien ook het resultaat dat ik verwacht had.

Het dimensioneel model voert sneller leesresultaten uit in vergelijking met het Data Vault, dit omdat het Data Vault model meer relaties heeft. Hierdoor zullen veel meer joins moeten gebeuren wanneer alle data moet opgehaald worden.

Bij de Data Vault methodologie wordt er meer informatie (Hash keys, informatie over extractie, ..) en tabellen opgeslagen in een databank. Dit zorgt ervoor dat de volumes van data enorm stijgen. Bijgevolg zal er dus een hogere kostprijs zijn om deze data te stockeren. Aangezien dit geen requirement is voor DHL Pharma Logistics, zou dit leiden naar een onnodige meerkost voor dit project.

De KPI's waarvoor een model moet opgesteld worden zijn gestandaardiseerd, en dienen niet flexibel te zijn. Indien de berekening voor de KPI's zouden gewijzigd worden, hoeven enkel sommige parameters uit het ETL-proces aangepast te worden en niet het datamodel zelf.

Het grote nadeel voor het implementeren van een project aan de hand van het dimensioneel model, is dat er niet kan gewerkt worden via de agile manier.

Er kan verder onderzoek verricht worden naar hoe het Data Vault model omspringt met

NoSQL-data en big data in het algemeen. Is het een betere keuze om een Data Vault model te gebruiken bij big data modellen (aangezien Data Vault parallel inladen van data toelaat)?

A. Onderzoeksvoorstel

Het onderwerp van deze bachelorproef is gebaseerd op een onderzoeksvoorstel dat vooraf werd beoordeeld door de promotor. Dat voorstel is opgenomen in deze bijlage.

A.1 Introductie

Wanneer het management in het bedrijf een strategische of tactische beslissing wil maken, is deze beslissing gebaseerd op data afkomstig uit verschillende databronnen. Daarom is er bij grote ondernemingen (en tegenwoordig ook bij KMO's) nood aan een rapporteringssysteem. Voor het opstellen en onderhouden van datawarehouses wordt een bepaald budget voorzien. Relaties leggen tussen verschillende data is dan ook een grote uitdaging. Daarom is het dus belangrijk dat het model op de juiste manier ontworpen wordt om kosten te beperken wanneer men de datawarehouse wil onderhouden/uitbreiden. Hiervoor bestaan verschillende modelleertechnieken. In dit onderzoek worden enkel het Kimball dimensioneel modelleren en Data Vault 2.0 vergeleken. We proberen in dit onderzoek de volgende vraag te beantwoorden: **Waar zitten de verschillen bij het modelleren met Data Vault 2.0 en het dimensioneel modelleren?**

Ook zal er een antwoord trachten gevonden te worden op volgende deelvragen:

- Zijn er verschillende manieren van aanpak mogelijk?
- Hoe flexibel/schaalbaar zijn beide systemen?
- Is er een verschil in performantie?
- Hoe verschillen de technieken naar onderhoud toe?

Bij DHL Pharma Logistics gebeurt het berekenen van de KPI's (Key Performance In-

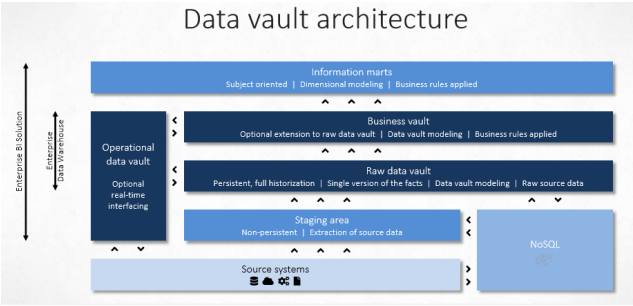
dicators) nog altijd manueel. Zo worden de KPI's berekend via een rekenmachine en handmatig ingevoerd in een Excel-bestand. De informatie die nodig is om verschillende berekeningen te maken is afkomstig uit verschillende databronnen (mainframe, Cronos, Excel-sheets, ..). Dit neemt veel tijd in beslag, dus beslist de firma om een deel van de KPI's te automatiseren. Hiervoor zal een datawarehouse moeten opgezet worden. Deze zal gemodelleerd worden in Data Vault 2.0. Maar is dit wel de beste oplossing? Dit onderzoek zal uitwijzen of Data Vault 2.0 wel degelijk de beste oplossing is.

A.2 Literatuurstudie

A.2.1 Data Vault 2.0

Architectuur

De architectuur van Data vault bestaat voornamelijk uit 3 lagen: De staging area, de raw data vault area en de Business vault. De staging area wordt gebruikt om alle data tijdelijk te stockeren. Daarna wordt de data doorgezonden naar de volgende laag: de raw data vault. Hierbij wordt de architectuur omgevormd naar een data vault. Hierop worden dan data manipulaties gemaakt en wordt de data doorgezonden naar de volgende laag, de business vault. Data marts worden gecreëerd op de business vault (Linstedt, 2015).



Figuur A.1: Data Vault architectuur voorgesteld door Stroobants (2018).

Entiteittype's

Bij data vault wordt er een onderscheid gemaakt tussen 3 verschillende entiteiten: hub, link en satelliet. In een hub wordt een hash sleutel opgeslagen die gebaseerd is op de identifier van die entiteit en metadata (zoals de bron en wanneer de record is ingevoerd). Een link is entiteit die verantwoordelijk is om verschillende hubs met elkaar te verbinden. Hierin worden de hash sleutels van de verbonden entiteiten in opgeslagen. Satellieten kunnen verbonden worden met hubs en links. Deze bevatten de inhoudelijke data van de entiteit.

A.2.2 Dimensioneel modelleren

Architectuur

Bij het dimensioneel modelleren via Kimball is er 1 enkele laag, hierin worden alle operaties uitgevoerd (ETL: Extraction, Transaction en Load). De data wordt ingeladen in een ster-schema. Op deze laag worden dan data marts gebouwd. (Jukic, 2006)

Entiteittype's

Bij deze techniek bestaan er 2 entiteittype's: feit tabellen en dimensionele tabellen. De feit tabellen bevatten alle transactionele data, data waarop je eigenlijk berekeningen kan maken. Dimensionele tabellen bevatten meer informatie over de transactionele data.

A.3 Methodologie

Voor dit onderzoek zullen er twee datawarehouses opgezet worden in een SAP HANA-omgeving. De eerste datawarehouse zal gemodelleerd worden in Data Vault 2.0, de andere in een dimensioneel model. De SAP HANA omgeving is 'on-premise' die draait in een Microsoft Azure omgeving. Het modelleren zal deels gebeuren in Eclipse (die een remote-verbinding maakt met Azure) en deels via een web IDE voor HANA (Xsengine). Wanneer beide datawarehouses operationeel zijn, kan er gestart worden met de vergelijking. De datawarehouses zullen gebaseerd zijn op KPI's die gedefinieerd zijn bij DHL Pharma Logistics.

A.3.1 Performatie

Om de performantie van beide systemen te vergelijken, zullen er een aantal verschillende queries uitgevoerd worden op data marts gebaseerd op deze datawarehouses. Op basis van uitvoeringstijd kunnen we deze dan met elkaar vergelijken. Zo kunnen we te weten komen of er wel degelijk een verschil is tussen beide architecturen in performantie en hoe groot de verschillen zijn.

A.3.2 Audit

Stel dat er op 2 verschillende databronnen klantengegevens opgeslagen wordt, zal er een keuze moeten gemaakt worden. Van welke bron haal ik mijn gegevens? Indien er verschillende problemen optreden met data, willen we graag kunnen onderzoeken waar het probleem zich heeft voorgedaan. Hiervoor voegen we META-data toe aan de data die ons verteld waar en wanneer de data werd opgehaald.

A.3.3 Schaalbaarheid

Hoe wordt er omgegaan met grote hoeveelheden data in beide architecturen? Merken we hier een significant verschil? Zien we de uitvoeringstijden lineair/exponentieel stijgen?

A.3.4 Flexibiliteit

De vereisten voor rapportering verandert vaak bij bedrijven. Soms moeten KPI's worden toegevoegd, soms moeten deze gewijzigd worden. Maar wat als er databronnen in het bedrijfsnetwerk toegevoegd? Hoe gemakkelijk kunnen deze wijzigingen gemaakt worden in beide architecturen? Dit zullen we onderzoeken door een nieuwe KPI toe te voegen aan het systeem.

A.4 Verwachte resultaten

Op basis van het uitgevoerde onderzoek zullen we hiervan een resultaat kunnen opstellen. Ik verwacht dat beide technieken zijn voordelen en nadelen zullen hebben. Zo zal Data Vault 2.0 een modelleertechniek zijn die zeer flexibel is, maar dit zal ten koste gaan van de performantie. Het dimensionele model zal zo performanter zijn, maar weinig flexibiliteit bieden.

A.5 Verwachte conclusies

Aangezien Data Vault 2.0 veel flexibiliteit te bieden heeft, zal dit de beste oplossing zijn wanneer alle data verspreidt staat op verschillende systemen. Bij Data Vault 2.0 is het namelijk mogelijk gemakkelijk nieuwe databronnen toe te voegen in een datawarehouse. Maar wanneer men de de data marts wil ontwerpen, zal men nog steeds moeten gebruik maken van dimensioneel modelleren. Wanneer een bedrijf weinig databronnen heeft en deze weinig veranderen, is dimensioneel modelleren de betere oplossing.

Voor DHL Pharma Logistics zal Data Vault 2.0 dan ook de beste oplossing zijn, aangezien hun data verspreidt staat over enkele systemen. Zo kunnen ze hun KPI's ook nog beter definiëren en makkelijker aanpassen in het systeem.

Bibliografie

Buneman, P. (1997). *Semistructured data*. University of Pennsylvania.

Gartner. (2019). *Magic quadrant for analytics and business intelligence platforms*. Gartner.

Gupta, A., Harinarayan, V. & Quass, D. (1995). *Aggregate-query processing in data Warehousing environments*. IBM Almaden Research Center en Stanford University.

Helfert, M., Zellner, G. & Sousa, C. (2002). *Data quality problems and proactive data quality management in data-warehouse-systems*. University of St. Gallen en University College Dublin.

Inmon, W. H. (2005). *Building the Data Warehouse*. Wiley Publishing Inc.

Inmon, W. & Linstedt, D. (2015). *Data Architecture: A Primer for the Data Scientist* (K. Herbert, Red.). Morgan Kaufmann.

Jovanovic, V. & Bojicic, I. (2012). *Conceptual Data Vault Model*. Georgia Southern University en Belgrade University.

Jukic, N. (2006). Modeling strategies and alternatives for dat warehousing projects. *Communications of the ACM*.

Kimball, R. & Ross, M. (2013). *The data warehouse toolkit: Third edition*. Wiley.

Kortink, D. L. M. M̃. A. (2000). *From enterprise models to dimensional models: a methodology for data warehouse and data mart design*. University of Melbourne.

Krneta, D., Jovanović, V. & Marjanović, Z. (2014). A direct approach to physical Data Vault design. *Computer Science and Information Systems 11*.

Langseth, J., Vivatrat, N. & Sohn, G. (2005). *Schema and ETL tools for structured and unstructured data*. Clarabridge.

Linstedt. (2015). *Building a Scalable Data Warehouse with Data Vault 2.0*.

Linstedt, D. & Olschimke, M. (2016). *Building a scalable data warehouse with data vault 2.0* (A. Invernizzi, Red.). Todd Green.

McAfee, A. & Brynjolfsson, E. (2012). Big data: the management revolution. *Harvard Business Review*.

Satyanarayana, R. (2010). Data warehousing, data mining, OLAP and OLTP technologies are essential elements to support decision-making process in industries. *International Journal on Computer Science and Engineering*.

Stroobants, J. (2018). Modern data warehousing with data vault in SAP HANA.

Vassiliadis, P. (2000). *Data warehouse modeling and quality issues* (proefschrift, National technical Univesity of Athen).