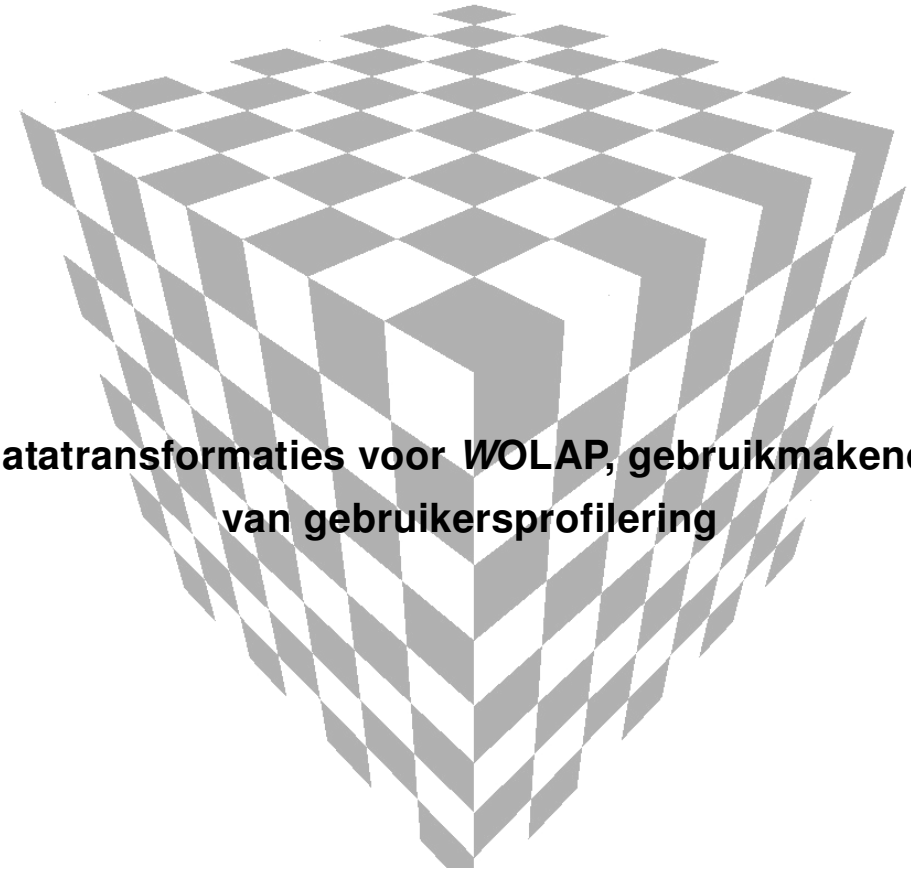
efficiency **online**



Datatransformaties voor *W*OLAP, gebruikmakend van gebruikersprofilering

Datum, Scriptienummer
maart 2005, 538

Auteur
Wouter Ewalds

Onder begeleiding van
Patrick van Bommel (RU) & Daniel van der Wallen (Efficiency Online B.V.)

Voorwoord

Na zes en een half jaar is het bijna het einde van mijn studie Informatica te Nijmegen.
In de eerste jaren heb ik me vooral gericht op bedrijfsgerichte informatica. Dit heeft zich ontwikkeld naar een grote interesse voor het thema Internet en veel bijbehorende aspecten.
Terugkijkend op deze periode, besef ik me hoeveel ik de afgelopen jaren geleerd heb niet alleen naar aanleiding van mijn studie, maar ook tijdens de nevenactiviteiten en mijn bijbaantjes.

Tijdens de studie ben ik ook actief geweest in het bestuur van de studievereniging Thalia, in het bestuur van de stichting Beet en in de organisatie van het Beestfeest. Hiervan heb ik veel geleerd in het kader van organiseren, plannen en presenteren. Ook was dit een zeer gezellige periode waar ik met een brede glimlach op terugkijk.
Naast de nevenactiviteiten heb ik tijdens de studie ook een bijbaan gehad bij een tankstation en zelfs de kans gekregen om via het GiPHouse te gaan werken in de I.T. sector, bij ENG Efficiency B.V. Hier heb ik ongelooflijk veel geleerd. Hier is mijn interesse ontstaan in het onderwerp van deze scriptie. Toen na 1.5 jaar ENG Efficiency B.V. overging in Efficiency Online B.V. kwam de aansluiting met het thema Internet tot stand.

Toen het einde van mijn studie in zicht kwam en de tijd rijp was om te gaan afstuderen hoefde ik niet lang te zoeken naar een onderwerp en een bedrijf die mij een stage kon verzorgen. Ik en Efficiency Online bleken allebei interesse te hebben in **O**n**L**ine **A**nalytical **P**rocessing (*OLAP*) en de toepassing hiervan via het **W**orld **W**ide **W**eb (*WWW*). Efficiency Online heeft kennis in huis op het gebied van zowel formele informatica en artificiële intelligentie waardoor er een goed leertraject mogelijk was. Ik heb hier mijn stage in gezellige, rustige sfeer kunnen uitvoeren zonder dat mijn werk in conflict kwam met de stage.

Rest mij nog een aantal mensen te bedanken voor hun inzet, hulp en steun. Aangezien bedanken vergeten met zich meebrengt, bedank ik hier de voor mijn stage belangrijkste personen in naam. Allereerst wil ik Patrick van Bommel en Daniel van der Wallen bedanken voor hun hulp, meedenken en vinger aan de pols houden wat betreft de voortgang. Vooral Patrick van Bommel heeft me vaak tijdig op het matje geroepen. Ik wil Sjoerd Geraedts en Daniel van der Wallen bedanken voor het introduceren van het onderwerp, voor de steun en voor de mogelijkheid mezelf te ontwikkelen in het bedrijf. Ik wil Stefanie Mudder bedanken voor alle steun en opbeuringen die ze me heeft gegeven tijdens deze periode. Daarnaast wil ik iedereen bedanken die me de afgelopen jaren heeft geholpen door het geven van steun, collegialiteit, vriendschap of op welke andere manier dan ook.

Inhoudsopgave

1. Inleiding 8
1.1 Efficiency Online 8
1.2 Begeleiding 8
1.3 Over deze scriptie 8

2. De opdracht 10
2.1 Introductie (Abstract) 10
2.2 Omgeving 11
2.3 Specificatie 11
2.4 Behoefte 14
2.5 Inzicht 14
2.6 Afbakening 14
2.7 Probleemstelling 15
2.8 Transformatie 16

3. OLAP 17
3.1 Inleiding 17
3.2 Standaard OLAP functionaliteit 19
3.3 Standaard OLAP transformatie 20
3.4 3-deling 23
3.5 Profilering 25

4. Grafen, Tabellen & Transformatie Methode 27
4.1 Grafen 27
4.1.1 Graaf transformaties 27
4.2 Database & Tabellen 28
4.3 Representatie 30
4.4 Transformatie Methode voor Tabellen 33
4.5 Inzicht 35

5. Het model 36
5.1 Inleiding 36
5.1.1 Algoritme 37
5.1.2 Algoritme, elementair 38
5.2 Heuristiek 43

6. Implementatie 44
6.1 Inleiding 44
6.2 Objectbeschrijving *Datamanager* 46
6.3 Proof of concept *WOLAP* tool 47
6.4 De datamanager 49
6.5 Testen 57
6.5.1 Testsets 57
6.5.2 Testresultaten 59

7. Conclusie 69
7.1 Op te leveren 69
7.2 De centrale probleemstelling 69

8. In de toekomst 70

Literatuurlijst 71
Bijlage A, Documentatie van de implementatie 73

1. Inleiding

In dit document zal beschreven worden hoe ik mijn afstudeerstage bij Efficiency Online B.V. (Later aangeduid als Efficiency Online) heb aangepakt. Deze afstudeerstage heeft plaatsgevonden in de periode september 2004 tot maart 2005. Dit document is een samenvatting van het hele afstudeerproces en de resultaten hiervan.

1.1 Efficiency Online

Efficiency Online levert producten en diensten op het gebied van **C**ustomer **R**elationship **M**anagement (*CRM*), financieel beheer, **H**uman **R**esource **M**anagement (*HRM*), project management, kennismanagement en data analyse. Vooral de processen die gaande zijn binnen bedrijven, ondersteunen ze met software, die in nauwe samenwerking met de klant tot stand komt. Het bedrijf is gevestigd aan de Houtmankade in Amsterdam, maar bevordert het flexibel werken waar de omstandigheden dit toelaten. Hierdoor zijn er vele mogelijkheden je te ontplooien.

1.2 Begeleiding

Bij Efficiency Online ben ik begeleid door Daniel van der Wallen, mede oprichter en directeur van het bedrijf. Zijn achtergrond ligt in de artificiële intelligentie en hij heeft de nodige jaren ervaring in de ICT industrie. Vanuit deze achtergrond zal hij mijn voortgang bewaken en me waar nodig, ondersteuning bieden.
Vanuit de Universiteit ben ik begeleid door Patrick van Bommel. Hij hield zich voornamelijk bezig met de ondersteuning en bewaking van de academische kant van mijn stage. Hij vormde tevens een belangrijke spil voor de bewaking van de voortgang, aangezien werken en stage lopen snel met elkaar kunnen conflicteren.

1.3 Over deze scriptie

Dit document is verdeeld in een aantal hoofdstukken waarin de gehele stage en het bijbehorende onderzoek beschreven worden. Allereerst wordt de opdracht besproken en de afbakening hiervan. Dit vormt de basis van de stage en deze scriptie.
Vervolgens worden alle belangrijke elementen geïntroduceerd om de lezer een idee te geven welke methoden en technieken er nodig zijn voor het uitvoeren van de opdracht.

Het model dat gegeven zal worden, is de box waarin methoden en technieken toegepast zullen worden in de implementatie. De implementatie zelf zal besproken worden in het gelijknamige hoofdstuk. Daarnaast zullen ook een aantal test cases genoemd worden om te toetsen of het model en bijbehorende box voldoen aan de eisen van de opdracht.

Aan het einde van deze scriptie zal ik ingaan op de resultaten, het verloop en de toetsing aan de opdracht in de vorm van een conclusie.

Rest hier nog te vermelden dat diagrammen zijn gemaakt op basis van cognitie met als doel verduidelijking door middel van beeldvorming. Als er een specifieke methode is gebruikt bij het maken van een diagram of schema, dan zal dit expliciet vermeld worden bij het betreffende figuur. Ik heb hiervoor gekozen om uit bepaalde specifieke methoden te kunnen stappen om een duidelijker beeld te kunnen schetsen, zonder hierbij in te leveren aan precisie.

2. De opdracht

In dit hoofdstuk wordt de opdracht besproken. Naar aanleiding van een bekend voorbeeld wordt langzaam toegewerkt naar de gebruikte technieken, de toepassingsgebieden en de centrale probleemstelling van deze scriptie. Kortom het hoe, wat en waarom.

2.1 Introductie (Abstract)

Stel je voor dat je vaste klant bent bij een ontzettend grote videotheek. Aangezien de videotheek een groot assortiment heeft, kan het even duren voordat je een keuze hebt kunnen maken. Dit is een probleem voor de videotheek en voor jou als klant.

De videotheek houdt echter bij wat je huurt en plaatst je daarom in een groep van klanten die ongeveer hetzelfde huurgedrag hebben. Vanuit deze groepen hebben ze een selectie gemaakt met de meest gevraagde films per groep. Op het moment dat je de videotheek binnenkomt, wordt je videotheekpas gescand. Vervolgens krijg je een lijst van nieuwe films die bij jou voorkeur aansluiten.
Daarnaast kijken ze per groep of ze het filmaanbod niet makkelijker aan jou, de klant, kunnen aanbieden. Een folder met alle films is niet leesbaar voor jou als klant. Je wilt immers alleen de informatie krijgen van de films waar je geïnteresseerd in bent, maar als het moet wil je de informatie kunnen achterhalen van alle films. Daarom deelt de videotheek bij de pasjesscanner folders uit met hierop de films waarvan ze weten dat je er geïnteresseerd in bent.
Naar mate je vaker de videotheek bezoekt, zal de wachttijd en de zoektijd minder lang worden. Zo verlaat je, als tevreden klant, het pand.

Deze introductie geeft aan waarover deze scriptie gaat. In het onderzoek heb ik geprobeerd een analyse methode (*OLAP*) toegankelijker te maken voor het *WWW*, door gebruik te maken van informatie over gebruikers en door na te denken over de verschillende methoden om de analyse methode aan te bieden en te implementeren.

Het algemene idee bestaat al, dit zul je misschien herkennen van webstores zoals bol.com en amazon.com, in de vorm van aanbevelingen. De analyse applicatie naar het *WWW* migreren in combinatie met de profilering is een nieuwe techniek, deze transformatie zal worden onderzocht, geïmplementeerd en getest. Het idee achter de scriptie is een samensmelting van bestaande technieken mogelijk maken door middel van transformaties en dit toegepast op een analyse applicatie, *OLAP*. Kortweg betekend dit dat met het gebruik van bestaande technieken *OLAP* aangeboden kan gaan worden op het *WWW*.

Wat *OLAP* nu precies voorstelt, inhoudt (hoofdstuk 3) en hoe de samensmelting tot stand zal komen (hoofdstuk 5) zal uitgebreid beschreven worden in dit document.

2.2 Omgeving

Vrijwel iedere organisatie heeft tegenwoordig een website. Zeker de laatste jaren is het steeds belangrijker geworden om de gebruiker snel de voor hem/haar relevante informatie aan te bieden. Dit proces kan gemeten worden door bepaalde statistieken bij te houden en hierover uitspraken te doen (denk hierbij bijvoorbeeld aan het aantal bezoekers per dag per continent van een website). Dit is slechts een voorbeeld van wat je zou kunnen analyseren. Het onderzoek abstraheert hiervan.

Het analyseren zou met behulp van een *OLAP* tool gedaan kunnen worden. De medewerkers van de organisatie zouden dit steeds vaker via het *WWW* willen doen (verderop aangeduid met *WOLAP*). Nu wil het feit dat *OLAP* een zware toepassing en het *WWW*, in essentie, een traag medium is. Om verwarring te voorkomen; het woord 'Online' in de term *OLAP* betekent in deze context niet hetzelfde als via het *WWW* 'Online' zijn. In *OLAP* betekent Online: koppeling.
Het is via een koppeling met een 'centrale' computer en koppelingen met meerdere gegevens mogelijk om analyses uit te voeren. De toevoeging *W* aan *OLAP* wil dus zeggen dat je de bestaande techniek beschikbaar maakt voor het *WWW*. Een uitgebreide uitleg over *OLAP* volgt in hoofdstuk 3.

2.3 Specificatie

Globaal zou je de opdracht samen kunnen vatten in: "zorg ervoor dat gegevens snel kunnen worden geanalyseerd via het *WWW* zonder dat je van te voren weet wat een gebruiker wil weten." Nu is *OLAP* een uitstekende techniek hiervoor, alleen geeft een directe vertaling naar het web performance problemen. Hier ligt de kern van de opdracht. Om ervoor te zorgen dat gebruikers snel de informatie kunnen bereiken die ze zoeken, is het idee geboren om gebruikers profilering toe te passen. In een applicatie worden de acties (analyses, algemeen gebruik) van een gebruiker bijgehouden. Op basis van deze informatie kun je profielen creëren.
Er is gebruik gemaakt van een gedegen voorspelling. Helaas liet de stage en afstudeeropdracht het qua tijd niet toe een test case uit te voeren met Safarii[1], wat een verrijking zou zijn geweest van het element profilering (hoofdstuk 8).

Dit houdt voor de opdracht concreet in:

- Grote hoeveelheden ruwe (niet geoptimaliseerde) data gebruiken.
- Het profileren van gebruikers.
- Het transformeren van een gedegen voorspelling van bovenstaande punt, naar een voor *OLAP* geschikt formaat.
- Een analyse applicatie ontwikkelen die hiervan gebruik kan maken en zich aanpast aan de gebruiker. Dit is de datamanager geworden die ook de verantwoordelijkheid heeft gekregen over de daadwerkelijke transformaties.
- Documentatie voor zowel gebruikers als systeembeheerders.

[1] een datamining tool die is ontwikkeld door Kiminkii (http://www.kiminkii.nl) in samenwerking met Dantec (http://www.dantec.nl/). Dantec is in 2003 met E.N.G. Efficiency B.V. samengegaan in Efficiency Online

Aan dit systeem, het geheel van alle componenten, is een aantal voorwaarden verbonden:

- De uiteindelijke analyse zal per gebruiker, na vergaring van gegevens over deze gebruiker even snel of sneller moeten gaan dan de normale *OLAP* techniek. Dit is tevens een van de doelen van de opdracht.
- De gebruikte hoeveelheid data moet representatief zijn voor het midden en kleinbedrijf, zodat resultaten direct de echte werkomgeving van Efficiency Online representeren. Dit dwingt ook schaalbaarheid af; je weet uiteindelijk hoeveel ruimte (in aantal tabellen) een transformatie met bijbehorend model gaat kosten. Met tabellen wordt hier verwezen naar tabellen in een SQL relationele database. Naast de benodigde processorcapaciteit zal er ook gekeken moeten worden naar de benodigde schijfruimte voor de datasets. Dit wil zeggen dat het aantal tabellen niet mag gaan exploderen. Dit brengt een grote beperking met zich mee. Het is dus niet mogelijk om lukkraak gegevens voor te berekenen.
- De consistentie van de data dient te allen tijde gewaarborgd te blijven (er mogen geen gegevens veranderen).
- De transformatie mag initieel lang duren maar moet incrementeel competitief zijn met standaard web applicaties. De transformatie moet gebruik maken van de normale *OLAP* techniek zodat het bijhouden van de data eenvoudig is en alleen het opzetten van een nieuwe analyse meer werk vereist.
- Het systeem zal robuust moeten zijn zodat er geen performance problemen optreden bij zwaar gebruik en zich geen onverwachte problemen voordoen. Bijvoorbeeld bij de transformatie van de data, mag de server niet overbelast of zelfs onbereikbaar worden. De data-integriteit wordt hierdoor ook deels gegarandeerd.
- Alle componenten die opgeleverd worden zullen in het Application Framework[2] dat Efficiency Online gebruikt voor ieder systeem moeten werken, er moet dus gebruik gemaakt worden van de API van dit Framework. Deze voorwaarde zal nog uitgebreid besproken worden in dit document. Voor een proof of concept kan hier eventueel vanaf geweken worden om conceptueel aan te kunnen tonen of een gemaakt keuze werkt.

Na de uitvoering van de opdracht zijn de volgende onderdelen opgeleverd (hoofdstuk 7):

- Een duidelijk omschreven basisidee en ontwerp (deze scriptie)
- Een model, de box, die aan bovenstaande voorwaarden voldoet
- Een implementatie van de datamanager
- Een proof of concept *WOLAP* applicatie, om de voorwaarden te kunnen testen en uitspraken te kunnen doen over het geheel. Hiermee wordt bedoeld dat er een verzameling analyses uitgevoerd zal worden op de modellen die gegenereerd worden.

Om een duidelijke scheiding aan te geven tussen al bestaande technieken en de nieuw te vormen techniek zal hieronder beschreven worden welke innovatie dit onderzoek probeert na te streven.

[2] Efficiency Online maakt gebruik van een Application Framework waar het geheel in zal moeten functioneren. Als vanzelfsprekend is het gebruik van dit Framework toegestaan

Er wordt gebruik gemaakt van de volgende technieken:

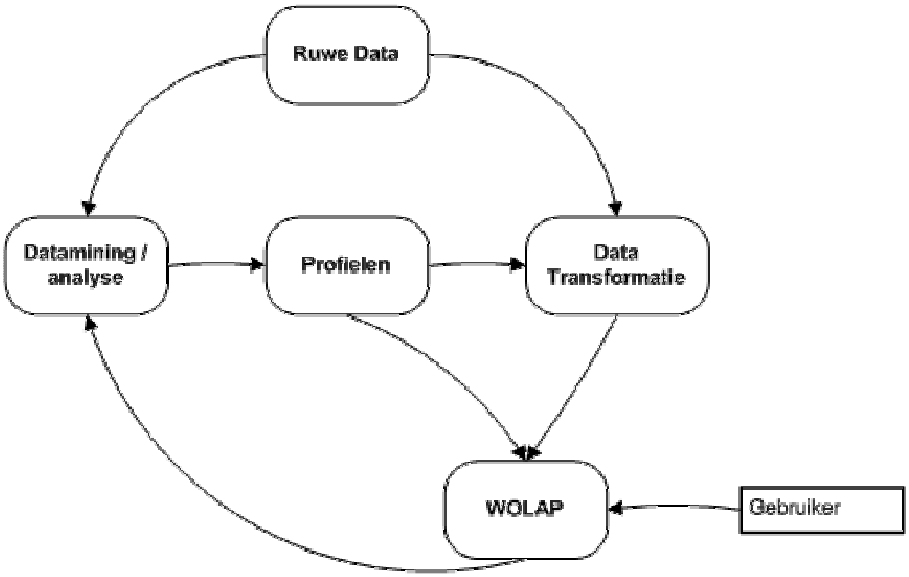
- *OLAP* transformatie
- *OLAP* functionaliteit
- Standaard profilering
- Kennis over het *WWW* met bijbehorende technieken

De innovatie die het onderzoek probeert te realiseren:

- Aangepaste *OLAP* transformatie voor *WWW* doeleinden, gebaseerd op kennis van het *WWW*, kennis over profilering en kennis van probleemdomeinen
- Aangepaste interfaces tussen componenten
- Door middel van zo min mogelijk overhead het Application Framework gebruiken voor het maken van de profilering en het gebruik hiervan in de transformatieslagen in het model

Zoals duidelijk af te leiden is uit bovenstaande informatie, zal de innovatie bestaan uit het combineren van kennis uit verschillende domeinen tot een nieuw product.

Bovenstaande informatie wordt schematisch weergegeven om een idee te krijgen van de benodigde componenten en de relaties hiertussen.



Figuur 2.3 *Globale, schematische weergave*

Zoals duidelijk in het plaatje te zien is, gaan we richting een hybride model (een combinatie van modellen en profielinformatie), waar meerdere technieken een rol zullen gaan spelen. De ruwe data is hier nog buiten bereik van een gebruiker gehouden, dit om alleen een globaal beeld te schetsen. Specifiekere stromingen en de uiteindelijke werking en relaties tussen componenten zullen uitgebreid besproken worden.

2.4 Behoefte

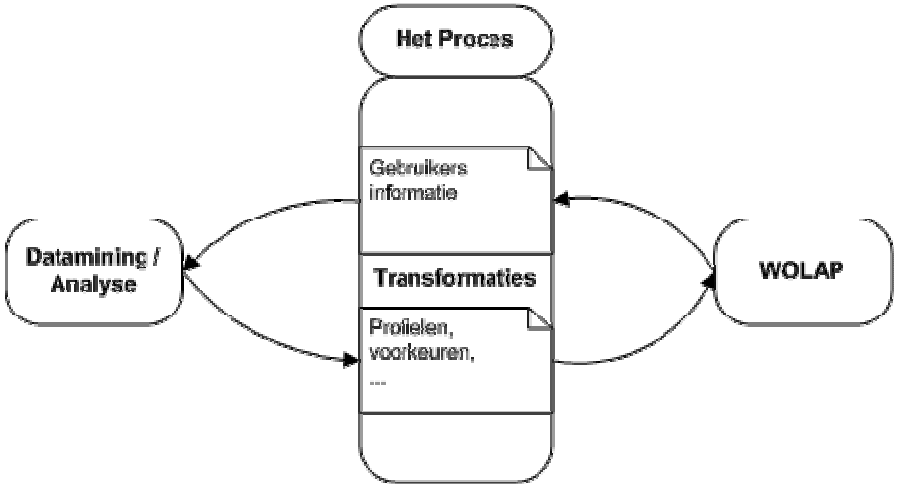
De behoefte van bedrijven om gecompliceerdere analyses dan bijvoorbeeld simpele webstatistieken online uit te voeren, is de afgelopen jaren enorm gestegen. Bedrijven als Microsoft en Oracle zijn momenteel bezig om de huidige intranet oplossingen om te zetten naar web oplossingen in respectievelijk .NET en de Oracle omgeving. Vaak zijn deze oplossingen te zwaar en financieel onaantrekkelijk voor het midden- en kleinbedrijf of kleinere afdelingen in grote bedrijven. Ook kan er een behoefte zijn om zich te onderscheiden van andere afdelingen, waarbij de verdere professionalisering van rapportages een belangrijk hulpmiddel kan zijn. Efficiency Online probeert aan deze behoefte te voldoen maar heeft nog geen tool om daadwerkelijk competitief te kunnen zijn.

2.5 Inzicht

De uitdaging ligt wat mij betreft geheel in het element profilering en de datatransformatie. Door deze methodes met elkaar te combineren probeer je een beter product op te leveren.
Men kan het zien als vals spelen. De eis “alle informatie moet opvraagbaar zijn”, zwak je af, door alleen de informatie die past bij het profiel van een gebruiker snel aan te bieden. Een gebruiker merkt hier idealiter niets van, waardoor het doel wordt bereikt. Althans, in zo verre dat je concessies moet doen richting systeemeisen en gebruikersaantallen.
Tevens wordt men bij het maken en onderzoeken van datatransformaties gedwongen om het probleemgebied tot in detail te doorgronden. De kennis en inzichten die tijdens deze fase worden opgedaan zijn zeer waardevol voor het ontwerp en implementatie van de *WOLAP* en transformatie tool.

2.6 Afbakening

Het gehele traject dat hier wordt beschreven is te uitgebreid om als geheel te implementeren binnen deze afstudeeropdracht. De focus ligt dan ook op de datatransformatie van de output van Safarii, of een schatting / andere methode hiervan, naar de input van een *WOLAP* applicatie (zie figuur 2.5). Het dataminen en de *WOLAP* applicatie kunnen als bekend worden verondersteld, al zal er aan de kant van *WOLAP* een proof of concept moeten worden opgeleverd omdat deze applicatie nog niet om kan gaan met de mogelijk andere output van een datatransformatie. Dit met de bedoeling de randvoorwaarden voor de transformatie al vervuld te hebben. Mocht de transformatie methodiek voorspoediger verlopen, dan is er altijd de mogelijkheid om daarna meer de focus te leggen op de *WOLAP* tool en het dataminen. Dit wil natuurlijk niet zeggen dat er niet gekeken wordt naar de functionaliteit van beide methoden; dit zal zeer belangrijk zijn om in het achterhoofd te houden. Het belangrijkste is dat er geen tijd verloren gaat met het ontwerpen of specificeren van methoden die niet van belang zijn voor de datatransformatie.



Figuur 2.6 *Het proces*

Uit bovenstaande striktere afbakening kan men afleiden dat het onderzoek als geslaagd kan worden gezien als er een goed ontwerp opgeleverd wordt met koppelingen naar zowel *WOLAP* als de profilering (Safarii / of andere methode). Tevens dient er een zogenaamd proof of concept opgeleverd te worden zodat te zien is of de transformatie afdoende is en om ervoor te zorgen dat implementatie aspecten meegenomen worden in het ontwerp.

2.7 Probleemstelling

Zoals is gebleken staat een aantal onderwerpen centraal:

- Profilering
- Data transformatie
- Data analyses (doorsnijdingen) door middel van WOLAP

Om deze verscheidenheid aan onderwerpen om te vormen tot een probleemstelling hebben we ervoor gekozen om het basisidee van de opdracht als leidraad te gebruiken.
Hiermee wordt de centrale probleemstelling:

> "Is het mogelijk om een algoritme te ontwikkelen dat via datatransformaties en datamining -gebruikmakend van een niet triviale gebruikersprofilering- uitvoer geeft die als invoer kan dienen om standaard *OLAP* functionaliteit efficiënter en sneller aan te kunnen bieden via het *WWW*."

Uit de probleemstelling kan worden afgeleid dat het daadwerkelijk gaat om het concept en zal een proof of concept van een *OLAP* applicatie voldoende zijn om aan te kunnen tonen dat de doelstelling bereikt is of kan worden bereikt.

2.8 Transformatie

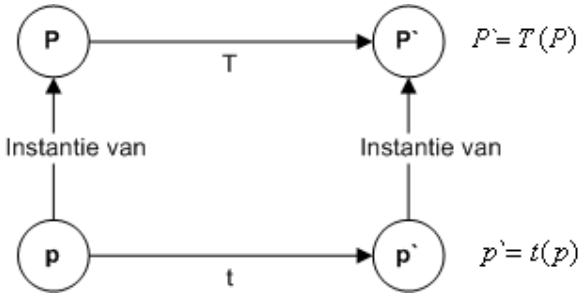
In deze scriptie staat het onderwerp transformatie centraal. In deze paragraaf zal algemeen beschreven worden, wat zo'n transformatie inhoud.

Een algemene definitie van een transformatie:

> "een transformatie is een relatie tussen twee programmaschema's P en P` (een programmaschema is een geparametiseerde representatie van een klasse van gerelateerde programma's; een programma van zo een klasse wordt verkregen door de schema parameters te instantiëren). Deze relatie blijkt correct als een zekere semantische relatie geld tussen P en P`."

Deze definitie is vrij vaag en algemeen. Om de definitie nuttig toe te kunnen passen, zal deze uitgewerkt worden.

Laten P en P` twee schema's zijn. Een schema kan ieder formeel beschreven entiteit bevatten. In deze scriptie bijvoorbeeld, zal een schema bestaan uit een database model bestaande uit één of meerdere tabellen. Een transformatie is de stap van P naar P`. Een transformatie is geldig als een voorgedefinieerde semantische relatie geldt. In figuur 2.8 is duidelijk te zien hoe een transformatie geïnterpreteerd dient te worden; P` wordt verkregen door een transformatie T toe te passen op P, men kan stellen dat P` het resultaat is van de functie T met P als argument.



Figuur 2.8 *een transformatie*

In het geval van deze scriptie wil dit zeggen dat, na het uitvoeren van een aantal stappen, het volgende optreed: P → P`. Deze implicatie heeft al een beperking in zich. Namelijk alles wat in P geldt, moet ook in P` gelden. Dit geldt overigens niet andersom. De relatie is niet equivalent, maar alleen implicerend. Een voorwaarde van een transformatie in deze scriptie is het behoud van gegevens, eventueel uitgebreid met extra gegevens geëxtraheerd uit de originele gegevens. Daardoor moeten alle semantische gegevens in P ook aanwezig zijn in P`. Het doel van de transformatie is snelheid te winnen en zo weinig mogelijk bruikbaarheid te verliezen.

3. OLAP

OLAP (OnLine Analytical Processing) is een veelomvattende term en wordt vaak als kreet gebruikt. Er is een definitie te geven die globaal aangeeft waar *OLAP* nu eigenlijk voor staat.

OLAP:

> "Zoekstrategie in databases waarbij gezocht wordt naar allerlei dwarsverbanden in gegevens."

Dit wil eigenlijk zoveel zeggen dat je uitspraken kan doen over gegevens en wel zo dat je alle aspecten die zich in de gegevens bevinden kan onderzoeken. Zo kan je bijvoorbeeld gegevens van bezoekers van een website bijhouden en hierna onderzoek gaan doen in deze gegevens. Iets abstracter; zie de gegevens als een kubus voor je; snij met een mes de kubus door; doe dit net zo lang, met eventueel stukken terugplakken, totdat je precies de gegevens (deel van de kubus) hebt die je wilt hebben.
In dit hoofdstuk zal worden uitgelegd hoe dit tot stand komt en zal de algemene problematiek van dit onderzoek besproken worden.

3.1 Inleiding

Laten we een simpele case beschouwen. Deze case is een simpel voorbeeld van website bezoeken. We willen uitspraken doen over de volgende bezoekersgegevens van een website. Iedere regel staat voor een zogenaamde hit, een vraag van een gebruiker aan een webpagina. Bijvoorbeeld door een pagina te openen of op een pagina op een link te klikken.

Gegevens weergegeven als tabel:

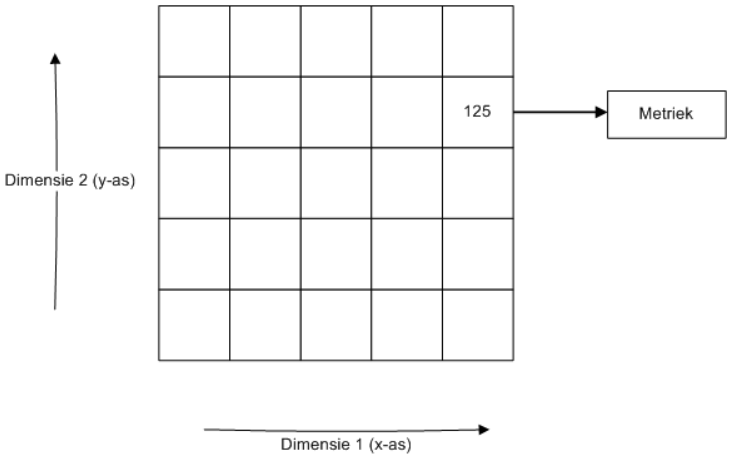
Id	Continent	Land	Jaar	Maand	Website
1	Europa	Nederland	2004	10	www.google.nl
2	Azië	Japan	2004	5	www.kun.nl
…	…	…	…	…	…

De tabel bestaat uit 6 kolommen en een aantal rijen. Aangezien in een tabel de volgorde van de kolommen en de inhoud van een kolom in iedere rij aan dezelfde syntax en semantiek voldoet, weet je precies waarover je uitspraken kunt gaan doen.
Een onderzoek, een view of doorsnijding genaamd, zou er als volgt uit kunnen zien:
"Het aantal bezoeken per website per jaar."

Iets abstracter zou je de denkbeeldige kubus éénmaal doorsnijden en geen stukken eraan plakken. Ga je kijken naar specifiekere gegevens bijvoorbeeld over een specifiek jaar dan snij je alle stukken van het eerder gesneden deel waar het jaar niet het specifieke jaar is eraf.

Als we van deze view[3] (figuur 3.1) abstraheren naar de losse elementen zien we het volgende:

1. een telling, de metriek genaamd
2. een soort van telling, de operator genaamd
3. 2 kolommen uit de gegevens tabel die je tegen elkaar wilt uitzetten. Deze worden dimensies of assen genoemd.



Figuur 3.1 *Grafische weergave van een view*

Kijken we nu opnieuw naar de gegevens zien we 6 dimensies die tegen elkaar uitgezet kunnen worden. Het aantal mogelijkheden voor een metriek is altijd gelijk aan het aantal dimensies. (Iedere dimensie kan dus 3 rollen vervullen: x-as, y-as of de metriek).
Het aantal operatoren dat mogelijk is hangt af van de eisen aan het *OLAP* programma dat je gebruikt.

Je kunt hier denken aan: het aantal, het aantal unieke, de som van, de vermenigvuldiging van, de deling van en dergelijke. Er zijn dus $6^3 \bullet n \geq 216 |$ (n = het aantal operatoren) mogelijke views te maken. Dit geeft aan hoeveel informatie er uit deze simpele dataset te halen is, wat niet wil zeggen dat dit altijd nuttige informatie zal zijn. Het belangrijkste is dat je de analyse gegevens die zich in de gegevens bevinden eruit moet kunnen halen.

Het is waarschijnlijk wel voor te stellen dat als er ontzettend veel gegevens, rijen en kolommen, zijn, dat het geheel dan aan de trage kant gaat worden. Zeker omdat je heel wat overhead hebt van het zoekprogramma zelf. De views moeten op een duidelijke, consistente manier weergegeven worden. Vandaar dat er wat optimalisatie gedaan moet worden. De methode hiervoor zal uitgebreid besproken worden in de rest van dit hoofdstuk.

In dit hoofdstuk zal aan bod komen wat de al bestaande technieken zijn die bruikbaar zijn om mee te nemen in de conversie naar het *WWW*. Als er concepten aan toegevoegd danwel gewijzigd zijn, zal dit expliciet vermeld worden.

[3] Een view is een doorsnijding van de data. Bijvoorbeeld bij web statistieken het aantal (operator) bezoekers (metriek) per jaar (x-as) per continent (y-as).

3.2 Standaard OLAP functionaliteit

Er kan gesteld worden dat via een *OLAP* tool altijd van iedere view naar iedere andere view gesprongen kan worden en wel zodanig dat de eerdere informatie behouden blijft of meegenomen wordt in een volgende view. Daarnaast is het belangrijk zoveel mogelijk af te leiden van een kolom met behulp van het type en de inhoud hiervan. Zo kun je bijvoorbeeld met een datum, 09-12-2004 11:41:38, veel meer dan alleen tellingen uitvoeren op deze datum. Dat is belangrijk in dit geval, omdat er weinig transacties of andere zaken precies in dezelfde seconden zullen gebeuren. Dit levert een groot aantal rijen op in de uitkomst waar je alsnog doorheen moet spitten. Door meer informatie uit een veld af te leiden, kun je dit spitten door de *OLAP* applicatie laten doen.

Uit het veld datum (*09-12-2004 11:41:38*) kun je bijvoorbeeld afleiden:

1. jaar, *2004*
2. maand, *12*
3. dag in de maand, *9*
4. week, *50*
5. dag in de week, *donderdag*
6. uur, *11*
7. minuut, *41*
8. seconde, *38*

In dit geval hebben we 8 dimensies gecreëerd uit 1 dimensie, zonder dat er iets aan de betekenis van het veld veranderd is. Om deze opsplitsing te kunnen benutten in de applicatie worden er relaties gedefinieerd om van een subdeel in te zoomen naar een volgend subdeel. Zo kun je bijvoorbeeld van jaar naar de maanden in dat jaar, van de maanden in dat jaar kun je weer naar de dagen van die maand van dat jaar etc. Deze functionaliteit heet Drill-Down.

Om het overzicht weer terug te krijgen (welke gegevens er ook weer allemaal waren bij een gegeven view), moet het mogelijk zijn om vanaf iedere willekeurige view uit te zoomen naar de originele gegevens. Deze functionaliteit heet Drill-Through.

Beide functionaliteiten dienen beschikbaar en uitvoerbaar te zijn in de proof of concept *WOLAP* tool en de data transformatie.

De overige triviale functionaliteiten worden bekend verondersteld.

3.3 Standaard OLAP transformatie

Voor *OLAP* is een standaard transformatie bekend. De transformatiemethode heet Starscheme[3]. Hieronder zal stapsgewijs dit proces weergegeven worden:

- Maak heuristieken om te bepalen welke kolommen relationeel worden en welke niet
- Bekijk bij iedere kolom of er nog meerdere dimensies uit af te leiden zijn, maak deze ook relationeel. (Dit wordt soms ook niet gedaan om het hele proces automatisch te laten verlopen, zonder tussenkomst van een gebruiker[4]).
- Onderzoek of deze heuristieken voldoen aan de te gebruiken database, pas ze zo nodig aan
- Onderzoek welke kolommen een zoekindex krijgen op basis van de te gebruiken database
- Maak het lege datamodel volgens de dataset
- Vul dit datamodel met de gegevens van de dataset. Hierbij mag geen informatie verloren gaan uit de data

Er zijn een aantal basisideeën voor dit algoritme:

- Hoe smaller een tabel hoe beter, dus ga je al snel vervallen in een relationele structuur
- Rekenen met integers gaat sneller dan met andere datatypen
- Iedere database heeft zijn eigen optimalisaties en methoden om deze te gebruiken

Deze ideeën geven weer waar bij de profilering van gebruikers op gelet dient te worden en zijn eisen voor de datatransformatie, losstaand van de profilering. De profileringsconcepten worden niet gebruikt in de huidige technieken.

Bij het opstellen van de heuristieken kan men al denken aan transformatie. Ik heb deze heuristieken strikt simpel gehouden om te voorkomen dat er bij de keuze van transformaties te snel gekozen zal worden voor een heuristische aanpak.

We kijken weer naar de gegevens uit het voorbeeld in dit hoofdstuk in paragraaf 3.1.
Er zitten twee mogelijke Drill-Down functionaliteiten in de dataset: van jaar → maand en van Continent → Land. Kijkt men verder, dan kan men ook de website onderverdelen in verschillende onderdelen, maar in dit voorbeeld abstraheer ik hiervan. Dit zal later dit hoofdstuk nog besproken worden.

Om hier even het onderwerp profilering aan te stippen wil ik een mogelijke profilering voorleggen. Hierdoor kan men een gevoel krijgen in de problematiek.

Mogelijke profileringen:

- Profilering 1:
  Groep geïnteresseerd in land, jaar en aantal hits
  Groep geïnteresseerd in alles wat er te halen valt
  …

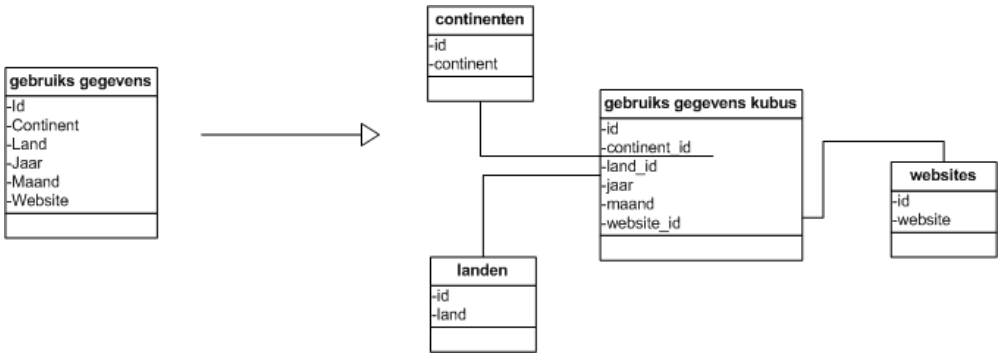
[4] De indeling in verschillende datamodellen zal in paragraaf 3.4 aan bod komen

- Profilering 2:
  Groep geïnteresseerd in de helft van alle dimensies
  Groep geïnteresseerd in alles wat er te halen valt
  …

Uit deze bewust gekozen profilering kan men de conclusie trekken dat er nog veel in te vullen is aan specifieke details over welke profilering nu het beste past bij *OLAP* en de datatransformatie. Er kan nu al wel worden geschat welke groep waarin interesse heeft.

Er zijn vele technieken om met profilering en generatie naar aanleiding van deze profilering om te gaan. We kiezen er echter voor om deze zelf te ontdekken met gebruikmaking van basisideeën die hun nut en werking al bewezen hebben. Ook hier kiezen we dus voor een hybride model. Deze manier van denken vinden we aansluiten bij het werken met het *WWW*.

Laten we gaan kijken naar de representatie van de gegevens en de Starscheme versie hiervan in een schema.



Figuur 3.3 *Origineel naar Starscheme*

We zien dat er drie kolommen zijn veranderd. Deze hebben een verwijzing (relatie) gekregen naar een externe tabel. Je kunt je indenken dat als je dit voor iedere kolom zou doen, dat je dan een ster vorm als schema krijgt. Vandaar de naam Starscheme.
De bron van de ster wordt ook wel kubus genoemd. Dit met het idee dat er vanaf iedere kant doorheen gesneden kan worden.

Conceptueel gezien zou je iedere kolom, behalve ‘id’, relationeel moeten maken. Als je naar de basisideeën kijkt, zou dit plausibel zijn. Voor dit voorbeeld heb ik stapsgewijs en naar redelijke maatstaven het proces dat hierboven gegeven wordt toegepast met het doel het principe duidelijk te maken. In hoofdstuk 6 wordt dit onderwerp uitgebreid besproken.

In tabelvorm ziet het er als volgt uit:

Origineel:

Id	Continent	Land	Jaar	Maand	Website
1	Europa	Nederland	2004	10	www.google.nl
2	Azië	Japan	2004	5	www.kun.nl
3	Europa	Nederland	2004	9	www.google.nl
…	…	…	…	…	…

Kubus:

Id	Continent_id	Land_id	Jaar	Maand	Website_id
1	1	1	2004	10	1
2	2	2	2004	5	2
3	1	3	2004	9	1
…	…	…	…	…	…

Continenten:

Id	Continent
1	Europa
2	Azië

Landen:

Id	Land
1	Nederland
2	Japan
3	Duitsland

Websites:

Id	Website
1	www.google.nl
2	www.kun.nl

Zoals te zien is, heb ik de kolommen, continent, land en website als te breed en te complex beschouwd, dus deze relationeel gemaakt. Door de nummers in deze kolommen te koppelen kun je de originele informatie weer terug vinden.

Uit de implementatie van het model is gebleken dat het relationeel maken van een kolom niet recht toe recht aan berekend kan worden. Er dient rekening gehouden te worden met de volgende zaken:

1. type van de kolom
2. maximale toegestane lengte van de kolom
3. werkelijke maximale lengte van de kolom
4. uniciteit binnen de kolom

Wat de invloed is van de interne representatie van de verschillende types binnen MySQL zal besproken worden in hoofdstuk 6.

In de volgende paragraaf zal de opdeling in verschillende transformaties worden besproken. De modellen die deze transformaties opleveren, worden aaneengekoppeld tot een hybride model, waar we gebruik maken van verschillende transformaties en de uiteindelijk verkregen profilering.

3.4 3-deling[5]

De in paragraaf 3.3 beschreven transformatie, is te zien als een 3-deling. Door gebruik te maken van verschillende transformaties kan men iedere gewenste functionaliteit aanbieden, zodat het gebruik van de data in een web applicatie gemakkelijk en snel aan te bieden is.
Dit klinkt vaag, in een tabel zal per methode aangegeven worden wat de voor- en nadelen zijn. Het doel is tot een hybride model te komen zodat er in het gebruik geen nadelen zijn voor ieder doel.

Model B is een tussenvorm die afgeleid is van de al bestaande techniek. Onderzoek heeft aangetoond dat het incrementeel opbouwen van dit model sneller werkt als het incrementeel opbouwen van Model C. Daarnaast is de overstap van model B naar C klein als men incrementeel opbouwen in ogenschouw neemt.

Het innovatieve idee is om van model A naar model C te gaan via model B, dit houdt de tussensprongen klein en de techniek beheersbaar voor het *WWW*.

Methode	Voordelen	Nadelen
1. Geen transformatie (Model A)	• Geen overhead bij gewoon gebruik (*insert* / *update* / *select*) • Maar 1 tabel nodig	• Doorsnijdingen maken is erg traag • Dimensie opdeling is niet mogelijk in de database omgeving • Reëel gezien niet te gebruiken voor *OLAP* functionaliteit
2. Starscheme transformatie, zonder extra opdeling in dimensies (Model B)	• Relatief gemakkelijk incrementeel bij te houden • Doorsnijdingen maken een stuk sneller dan bij een platte tabel • Voor kleinere datasets te gebruiken voor *OLAP* functionaliteit	• Overhead bij gewoon gebruik • Dimensie opdeling wordt hier niet gebruikt • 1 + #dimensies (zichtbaar in de originele dataset) tabellen
3. Starscheme transformatie (Model C)	• Goed te gebruiken voor *OLAP* functionaliteit • Snelste doorsnijding • Dimensie opdeling gaat zover de gebruiker wil	• Veel overhead bij gewoon gebruik • Moeilijker incrementeel bij te houden • 1 + #dimensies (zichtbaar in de originele dataset) tabellen

[5] In hoofdstuk 6 zal de tabel terugkomen om te verifiëren of deze ideeën kloppen met de resultaten van de implementatie. Dit zal niet hier besproken worden omdat de basisideeën nog niet geïntroduceerd zijn.

Zoals duidelijk in de tabel af te lezen is, zitten aan alle drie de vormen de voordelen die voor een web applicatie geschikt zijn, als er de nadrukkelijke wens bestaat om gebruik te gaan maken van *OLAP* in de applicatie. De nadelen worden nagenoeg opgeheven van elkaar.
Natuurlijk heeft het gebruik van deze 3 methoden als geheel als grote nadeel dat dit meer resources kost en ook meer tijd om ze synchroon te houden.

Experimenten hebben aangetoond dat in dit geval het onderhoud nagenoeg te verwaarlozen is, als men de volgende regels hanteert:

1. De stroom van data is als volgt:
    a. De data wordt real-time bijgehouden in model A
    b. De data wordt incrementeel bijgehouden in model B
    c. De data wordt incrementeel bijgehouden in model C, gebruikmakend van model B

2. Alle oude data blijft bestaan. Er zullen geen regels (rijen in een tabel) gewijzigd dan wel verwijderd worden.

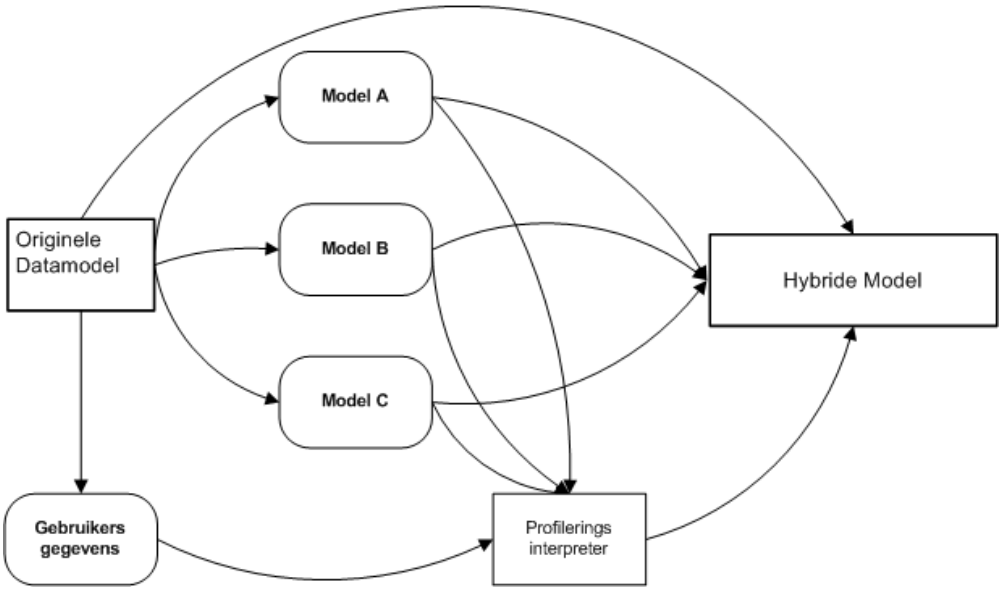
3. Men kan zelf het tijdstip kiezen waarop men B & C incrementeel wil updaten. Zo kan men zelf de live tijdstippen kiezen (het moment waarop alle modellen gesynchroniseerd worden). Bij iedere *insert* zou deze zelfs direct incrementeel gewijzigd kunnen worden maar dit zal ten nadele van de werking van een applicatie kunnen gaan. Bovendien kunnen zich dan datalock problemen gaan voordoen. Vandaar de eerste aanname (punt 1).

De driedeling die hierboven beschreven is zal getoetst worden aan de hand van de resultaten van de implementatie fase (Hoofdstuk 6).

3.5 Profilering

Tot nu toe is er vooral aandacht besteedt aan de bruikbaarheid in een bestaande applicatie en aan het kunnen aanbieden van *OLAP* functionaliteit in een applicatie. De profilering is tot nu toe buiten beschouwing gelaten.

De opzet van het geheel zal hier eerst schematisch uiteengezet worden.



Figuur 3.4 *Van origineel model naar een nieuw model met OLAP functionaliteit gecombineerd met gebruikers profilering*

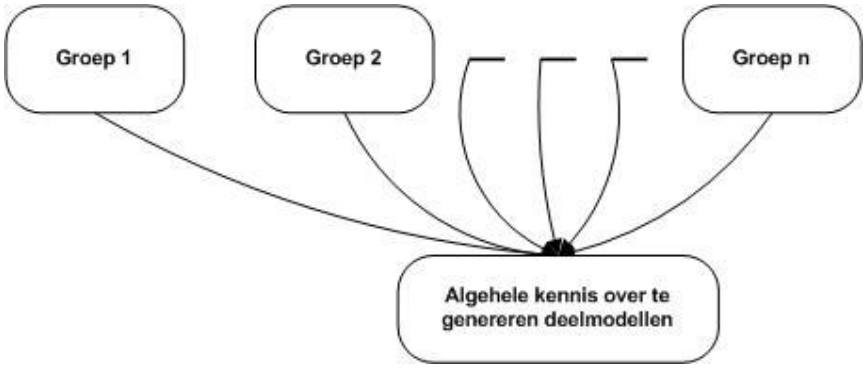
Het schema geeft aan dat in het nieuwe, complete model, ook de twee extra transformatie methoden en de voorberekende tabellen n.a.v. de gebruikers gegevens zitten. Deze gegevens bevatten alle soorten, dus niet alleen van het gebruik van *OLAP* maar ook van overige modules binnen een systeem. Deze kan men echter eenvoudig onderscheiden op basis van de dimensie kennis uit een bepaald model.

Deze gegevens kunnen worden opgeslagen door het bijhouden van parameters die met een aanvraag van een pagina worden meegezonden. De profileringsinterpreter zal vervolgens deze gegevens omzetten naar een lijst van voor te berekenen tabellen. Deze profileringsinterpreter kan bestaan uit een datamining tool of uit simpele regels op basis waarvan de lijst van voor te berekenen tabellen wordt gegenereerd.

We gaan er voor de rest van dit document vanuit dat de gebruikersprofilering verkregen wordt uit de gebruiksgegevens. De profilering wordt dus puur en alleen op basis van pagina aanvragen verkregen.

Veel gebruikte methoden om ook de permissies (welke groep, rol heeft een gebruiker) mee te nemen, laten we buiten beschouwing, omdat we n.a.v. alle gegevens één lijst (groep) gebruiken. Voor deze methode is gekozen omdat voorberekende tabellen voor iedereen beschikbaar moeten zijn.

We hebben gekozen voor group-sharing[7]. Onder water zullen alle opgevraagde gegevens afgehandeld worden op een zo snel mogelijke manier. Dus ook als een bepaalde groep een aantal voorberekende tabellen heeft, kan een gebruiker uit een andere groep deze tabellen benaderen. (Figuur 3.4.1)



Figuur 3.4.1 *Profileringsinterpreter*

De profilering in groepen wordt gebruikt voor de GUI (**G**raphical **U**ser **I**nterface) van de *WOLAP* tool. Hierdoor lijkt de profilering expliciet aanwezig, maar wordt deze onder water alleen gebruikt voor het opbouwen van een gemene deler van de dimensies. Hierbij kan men denken aan het initieel weergeven van dimensies en operatoren.

In hoofdstuk 5 zal uitgebreid ingegaan worden op het ontwerp van figuur 3.4.1. Om de problematiek weer te geven is ervoor gekozen om in het algemene hoofdstuk over *OLAP* het hybride model al te introduceren.

4. Grafen, Tabellen & Transformatie Methode

In dit hoofdstuk wordt de theorie besproken die gebruikt is tijdens het onderzoek. Om een idee te krijgen van hoe het model, de box, tot stand is gekomen en welke problemen je tegenkomt bij het maken van keuzen en het opstellen van de box, is dit hoofdstuk toegevoegd.
Zonder de theorie achter de probleemstelling te begrijpen zal de box, hoofdstuk 5, uit de lucht komen vallen en zal de achterliggende grondgedachte van graaftheorie te vaag zijn.

4.1 Grafen

Een definitie van een graaf[2].

Een graaf G bestaat uit 2 verzamelingen; L en P.

Voor P geldt:
- P is niet leeg
- Alle elementen uit P noemen we punten

Voor L geldt:
- Elk element uit L is een ongeordend paar van unieke elementen uit P
- $L = P * P$
- Alle elementen uit L noemen we lijnen

4.1.1 Graaf transformaties

Bij graaf transformaties wordt vaak gebruik gemaakt van een gelabelde, gerichte graaf[3].

Een gelabelde gerichte graaf G: $G = (P, L, bron, doel, label)$

Voor P geldt:
- P is niet leeg
- Alle elementen uit P noemen we punten

Voor L geldt:
- Elk element uit L is een ongeordend paar van unieke elementen uit P
- $L = P * P$
- Alle elementen uit L noemen we lijnen

Voor $bron$ geldt:
- Dit is het element dat bij iedere lijn een start punt aangeeft

Voor $doel$ geldt:
- Dit is het element dat bij iedere lijn het eind punt aangeeft

Voor $label$ geldt:
- Dit is het element dat bij ieder punt een label toewijst

Een graaf transformatie komt overeen met het toepassen van een regel of regels op een graaf en herhaling van dit proces totdat de regel of regels niet verder toegepast kunnen worden op de desbetreffende graaf.

Een regel[3]:

Een regel R: R = (L, R, V, lijm, verankering, condities)

Voor L, R geldt:
- Een linker en rechter graaf

Voor V geldt:
- We noemen dit de interface graaf
- V is een deelgraaf van L

Voor lijm geldt:
- Een voorkomen van V in R, relateert V met R

Voor verankering geldt:
- Verankeringsrelatie, relatie tussen punten uit L & punten uit R

Voor condities geldt:
- Toepassingscondities voor de regel

4.2 Database & Tabellen

In de praktijk werken veel applicaties met relationele databases en dus tabellen. Om het transformatie mechanisme toe te kunnen passen zal er een model moeten komen die de te gebruiken transformatie methode bevat en kan werken met tabellen.
Een eerste aanzet om dit te bereiken is het gaan onderzoeken van de mogelijkheden om een tabel te representeren in een graaf.

Allereerst dient gespecificeerd te worden wat men onder een database en zijn elementen verstaat.

Specificaties:

Een database DB is een verzamelingen tabellen; Tbl.
Een tabel Tbl is een geordende verzameling rijen; Rn, gedefinieerd door een type rij.
Een rij Rn is een geordende verzameling van $n \mid n \in \mathrm{N}, n > 0$ elementen, ook wel cellen genoemd; Cl.

Voor DB geldt:

- DB heeft de eigenschappen:
  - Gelabeld
  - Getypeerd
- $DB \equiv \{Tbl\}$

Voor Tbl geldt:

- Tbl heeft de eigenschappen:
  - Gelabeld
  - Getypeerd
  - Bevat een Rn_0.
  - Het aantal tabellen is theoretisch oneindig
- $Tbl \equiv \{Rn \mid Rn_0 = \{Cl^*\}\}$

Voor Rn geldt:

- Rn heeft de eigenschappen:
  - Heeft een vaste lengte gedefinieerd door het aantal Cl^*
  - Het aantal rijen is theoretisch oneindig
  - Bevat een Rn_0 die de volgorde van de cellen en de eigenschappen per cel beschrijft.
- $Rn \equiv \{Cl \mid Rn_0 = \{Cl^*\})$

Voor Cl geldt:

- Cl heeft de eigenschappen:
  - Heeft per Cl een beschrijving Cl*
  - Per Cl* is bekend:
    - Getypeerd
    - Gelabeld
    - Lengte
    - Default inhoud

4.3 Representatie

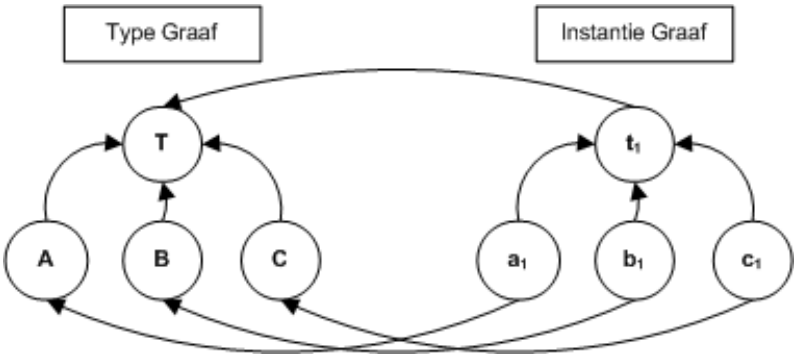
Het belang van het onderzoeken van representatie is groot. Hierdoor krijg je een beter beeld van de kracht van een transformatie en het terrein waarop je het wilt toepassen. Hier zal worden beschreven hoe een tabel als een graaf gerepresenteerd wordt.

Beschouw de volgende triviale tabel:

Tabel T:

A	B	C
A_1	b_1	c_1

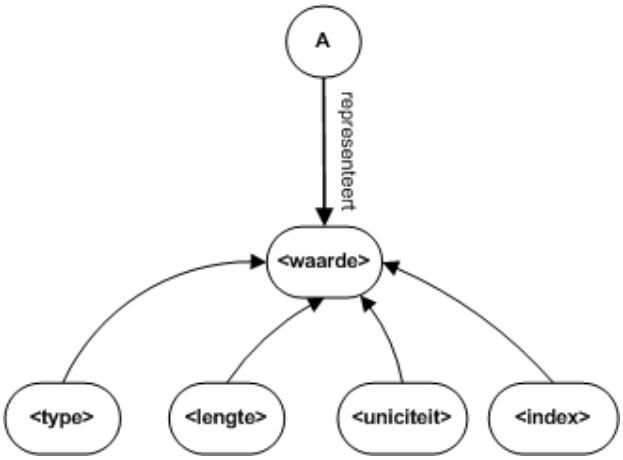
Tabel T is als volgt te representeren als een graaf:



Figuur 4.3.1 *Graafrepresentatie van een tabel [graaf]*

Deze representatie geeft de relatie tussen de verschillende elementen duidelijk weer. Zoals te zien is blijft de kolom indeling gegarandeerd intact, aangezien iedere instantie van A direct gekoppeld wordt aan zijn type. Ook kan een instantie (rij) niet aan een andere tabel gekoppeld zijn omdat een instantie (rij) direct gekoppeld is aan zijn type.
Zo op het eerste gezicht lijkt deze representatie geschikt te zijn om te gaan gebruiken, maar er zijn nog wel wat problemen die opgelost moeten worden.
Hoe geef je aan welk type een kolom heeft of hoe geef je de lengte of andere eigenschappen van een tabel weer in deze representatie?
Dit probleem is op te lossen door bij iedere instantie van een kolom en de typering hiervan te zorgen voor een typeringsgraaf die genest is in de kolomdefinitie zelf. Door de relatie die gelegd is tussen type en instantie zullen de eigenschappen vastgelegd zijn. Daarnaast wordt het label dat een instantie heeft de waarde, zodat het dynamische component van een kolom ook meegenomen wordt in de representatie.

Een kolom representatie ziet er dan als volgt uit:



Figuur 4.3.2 *Graafrepresentatie van een kolom [graaf]*

Beschouw de kubus van tabel T:

Tabel Tk:

A_id	B	C
A_id_1	b_1	c_1

Tabel As:

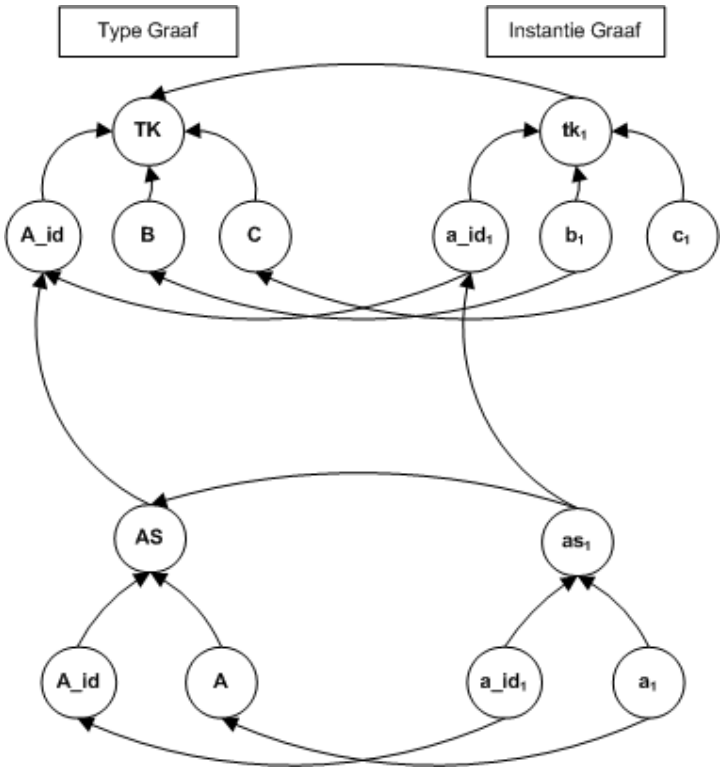
A_id	D
A_id_1	d_1

Dit is een triviaal voorbeeld om aan te geven hoe de graafrepresentatie tot stand komt.
We nemen dus aan dat kolom aan de voorwaarden voldoet om relationeel te worden opgenomen. Om dit in een graaf weer te kunnen geven, zal er allereerst een type graaf gedefinieerd moeten worden voor de relatie tabel, volgens de tabeldefinitie hiervan. Daarnaast zal een instantiegraaf de nodige koppelingen moeten krijgen met de typegraaf en vervolgens kan gekeken worden naar de relatie die gelegd gaat worden met de entiteit in de kubus graaf.

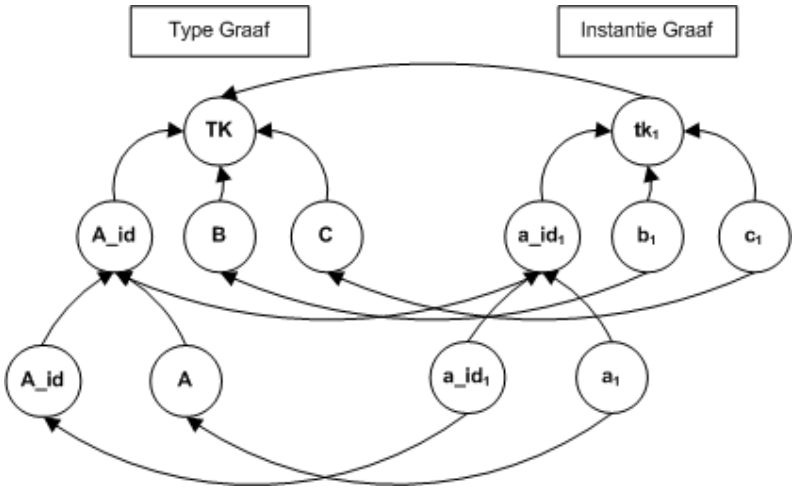
Nadat dit is uitgewerkt wordt er een generalisatieslag gemaakt waarin de expliciete definiëring wordt opgenomen in de kubus graaf.

Beide onderdelen zijn opgenomen in figuur 4.3.3 en 4.3.4.

Gerepresenteerd in een graaf zou dit er als volgt uit kunnen zien:



Figuur 4.3.3 *Graafrepresentatie van een kubus [graaf]*



Figuur 4.3.4 *Graafrepresentatie van een kubus na generalisatie [graaf]*

4.4 Transformatie Methode voor Tabellen

Na het proberen van verschillende methoden om een tabel te representeren als een graaf, om zo de graaf transformatie methode te gaan toepassen, bleek het handiger om het transformatie algoritme om te gaan schrijven voor tabellen, met behoud van kracht en functionaliteit. Aangezien je door middel van een algemene methode uitzoomt van het probleem kun je meer inzichten opdoen in de materie. Het bleek dat het graaf gericht denken de relatie tussen het specifieke standaard algoritme van het Starscheme en het omgaan en hanteren van tabellen duidelijk heeft gemaakt.

Het is nu ook vrij eenvoudig in te zien dat de algemene transformatie methode gemakkelijk is toe te passen op tabellen.

De transformatie methode omgeschreven naar tabellen:

Een regel[3]:

Een regel R: R = (L, R, V, lijm, verankering, condities)

Voor L, R geldt:

- Een linker en rechter tabel, of een verzameling tabellen gekoppeld dmv een relatie (foreign key).

Voor V geldt:

- We noemen dit de interface kolom, met de eigenschappen van een kolom zoals gegeven in figuur 4.3.2
- V is een kolom die voorkomt in L

Voor lijm geldt:

- Een voorkomen van V in R, relateert V met R

Voor verankering geldt:

- Verankeringsrelatie, relatie tussen een kolom of hun eigenschappen uit L & een kolom of hun eigenschappen uit R

Voor condities geldt:

- Toepassingscondities voor de regel

Dit algoritme is vrij algemeen. Hieronder volgt een concreet voorbeeld om duidelijk te maken hoe zo'n transformatie slag zou verlopen.

Beschouw de volgende tabel:

Origineel (L):

Id	Continent (V)	Jaar
1	Europa	2004
2	Azië	2004
3	Europa	2004
…	…	…

De kolom Continent komt in aanmerking voor een transformatie stap, vanwege de eigenschappen die overeenkomen met de eisen hieraan.
In tabel vorm komt het er dus zo uit te zien (R):

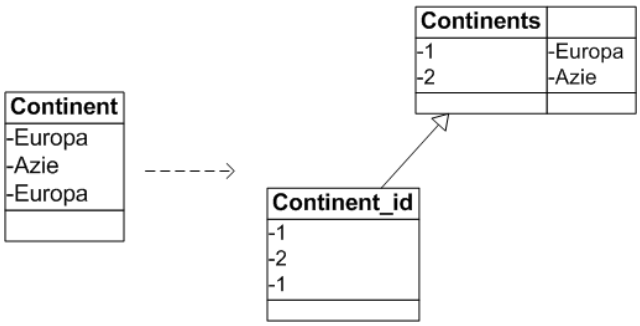
Kubus:

Id	Continent_id	Jaar
1	1	2004
2	2	2004
3	1	2004
…	…	…

Continenten:

Id	Continent
1	Europa
2	Azië

Schematisch kun je het zo zien dat een kolom vervangen wordt door een kolom en een tabel.



De lijm, verankering en condities zijn goed te begrijpen als je een mechanisme hebt die een kolom door een andere kan vervangen, de consistentie van de data kan garanderen en een algoritme bevat die bepaald of er transformatie plaats gaat vinden.

4.5 Inzicht

Door te werken met grafen abstraheer je gedachten van de onderliggende lagen. Met grafen zie je in welke mechanismen je toe kunt passen op tabellen. Door de graaftransformatie onder de knie te krijgen is het eenvoudig in te zien waar je voor moet waken als je een databasemodel gaat transformeren, namelijk triviale transformaties en het behoud van relaties tussen eenheden uit het databasemodel.

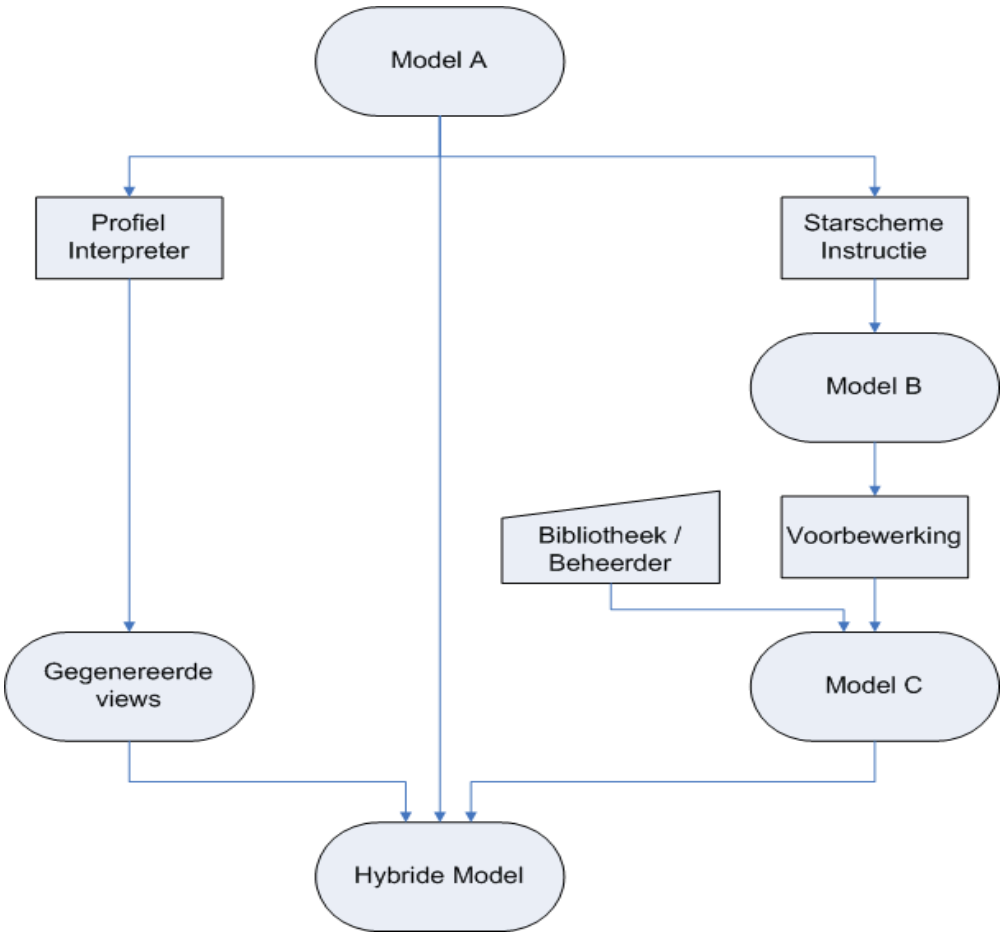
Door een tabel niet als semantisch object te beschouwen, maar als een entiteit met bepaalde eigenschappen krijg je gevoel voor de waarden van de eigenschappen en hun nut.

5. Het model

In dit hoofdstuk zal het model, de box, gegeven worden van de gewenste transformatie en bijbehorende invloeden van gebruikersprofilering. In het hoofdstuk “implementatie” zullen vervolgens alle bijbehorende datastructuren en datamodellen weergegeven worden.

5.1 Inleiding

Het model is een uitwerking van het gegeven model in figuur 3.4 en maakt gebruik van de besproken 3-deling in hoofdstuk 3, paragraaf 4. Om dit model daadwerkelijk uit te werken, wordt in paragraaf 5.1.1 een abstract algoritme gegeven en zal in paragraaf 5.1.2 dit algoritme elementair uitgewerkt worden. In paragraaf 5.2 zullen de openstaande concepten verder uitgewerkt worden om 5.1.1 en 5.1.2 puur functioneel te laten voor de algoritmen.
In figuur 5.1 is te zien wat de architectuur is van het hybride model en de flow van de stappen.



Figuur 5.1 *Top-Down architectuur van het hybride model [flowchart]*

5.1.1 Algoritme

In deze paragraaf zal het abstracte algoritme gegeven worden. Allereerst zullen de aannamen gedefinieerd worden:

Zij BrnTbl de bron tabel.
Zij UsrTbl de gebruikers tabel.
Zij AcTbl de tabel waarin acties van gebruikers worden opgeslagen. Deze acties zijn een deelverzameling van de mogelijke acties die door een systeem worden aangeboden.

De BrnTbl heeft de volgende eigenschappen:

- $n \in \mathrm{N}, kolommen$
- $m \in \mathrm{N}, rijen$

UsrTbl bevat minimaal de volgende gegevens:

- id
- gebruikersnaam
- wachtwoord

AcTbl bevat minimaal de volgende gegevens:

- id
- UsrTbl.id (Foreign Key naar UsrTbl)
- URI
- {systeeminformatie}

Een abstract algoritme:

1. Maak de modellen:
   i. Maak Model A
   ii. Maak Model B
   iii. Maak Model C

2. Op basis van de Modellen A,B,C & AcTbl:
   Maak Additioneel model

   Deze bestaat uit:
   i. Model A
   ii. Model B
   iii. Model C
   iv. Voorgegenereerde tabellen
   v. Alle overige tabellen

3. Bied het Additioneel model aan in het systeem als zijnde een *WOLAP* dataset.

5.1.2 Algoritme, elementair

Laten we nu het algoritme uitwerken.

Stap 1a
Om van het originele model naar Model A te gaan hebben we niets nodig. We houden alleen bij dat de BrnTbl de platte gegevens bevat.

Stap 1b
Het maken van Model B:

1. Analyse BrnTbl, dit levert een Starscheme instructie op.

   Zij $i \in \mathrm{N}$ een index die correspondeert met een kolom.
   Zij *instructie* [] een lijst, met *instructie* [*j*] een element uit deze lijst voor zekere $j \in \mathrm{N}$
   Zij *meetHeuristics ()* een functie die bepaalt of een kolom geëxpandeerd moet worden.
   Zij *createDimTbl ()* een functie die een lege dimensie tabel maakt

```
for (i=0; i < n; i++) {
        if (meetsHeuristics(i)){
                instructie[i] = 'expand';
                createDimTbl(i);
        } else {
                instructie[i] = 'keep';
        }
}
```

   Het algoritme geeft aan hoe een Starscheme instructie wordt opgebouwd. Het interessantste onderdeel, het bepalen of een kolom aan de heuristieken voldoet, zal in paragraaf 5.2 aan bod komen.
   Impliciet wordt in dit algoritme ook de tabel aangemaakt waar de getransformeerde data in opgeslagen zal worden. Op het moment dat je weet of een kolom aan de heuristieken voldoet, dus om een externe dimensie tabel op te leveren, weet je ook dat er een foreign key op de plaats van deze kolom komt te staan. Vanwege dit implementatie detail is dit niet expliciet opgenomen in het algoritme.

2. Transformeer BrnTbl op basis van de Starscheme instructie.

   Zij CbTbl de tabel waar de getransformeerde data in wordt opgeslagen.
   Zij $i \in \mathrm{N}$ een index die correspondeert met een kolom.
   Zij $j \in \mathrm{N}$ een index die correspondeert met een rij.
   Zij $instructie$ [] een lijst, met $instructie$ [*k*] een element uit deze lijst voor zekere $k \in \mathrm{N}$
   Zij row [] een lijst, met row [*k*] een element uit deze lijst voor zekere $k \in \mathrm{N}$
   Zij $getDimVal\ ()$ een functie die het id retourneert dat hoort bij waarde van de huidige kolom in de huidige rij. Bestaat dit id nog niet, maakt deze functie een nieuwe rij in de dimensietabel van de huidige kolom aan en retourneert de functie het id van de nieuwe rij.
   Zij $Val\ ()$ een functie die de waarde van de huidige kolom in de huidige rij retourneert.
   Zij $AddRowToTbl\ ()$ een functie die de opgebouwde rij toevoegt aan de tabel CbTbl.

```
for (j=0; j < m; j++) {
        for (i=0; i < n; i++) {
                if (instructie[i] == 'expand'){
                        row[i] = getDimVal(i);
                } else {
                        row[i] = Val(i);
                }
        }
        addRowToTbl(row);
}
```

   De functies die hier gebruikt worden zijn triviaal te noemen voor het algoritme. Deze zijn zo voor zichzelf sprekend dat dit is beschouwd als een implementatie detail.

In het algemeen zal opvallen dat dit een lineair algoritme is. Hieruit valt meteen af te leiden dat het waarschijnlijk niet de meest efficiënte manier is om de transformatie te implementeren. De resultaten van het onderzoek zijn vooral gericht op de bruikbaarheid van de getransformeerde data, het transformeren van de data zelf dient alleen incrementeel snel te zijn. Het is snel genoeg na te gaan dat het algoritme aan deze eis voldoet. Aangezien je per brontabel eenmaal de instructie opbouwt, heb je alleen nog de 2de fase van het creëren van model B nodig. Alleen als de tijdsspanne tussen de incrementele uitbouwfasen te groot wordt, is de kans groter dat het geheel te inefficiënt wordt. Dit is op te lossen door een caching systeem te laten draaien over de dimensie tabellen om het opzoeken van de id's zo snel mogelijk te maken.
Voor de doeleinden van Efficiency Online en voor mijn onderzoek moet het algoritme snel genoeg zijn.

Stap 1c

Het maken van Model C:

1. Voer de fasen 1 & 2 uit van stap 1b. Mochten deze al uitgevoerd zijn, sla deze dan over. Je maakt dus gebruik van het model dat stap 1b oplevert.

2. Om de nodige complexiteit te vermijden, ga ik bij deze fase een stapje hoger qua abstractieniveau. Het algoritme voor het extraheren van extra dimensies zal in natuurlijke taal en pseudo-code gegeven worden. De reden hiervoor is om overbodige formaliteiten achterwege te laten. De overbodigheid zit hem daarin dat de gebruiker (aanbieder van de dataset) die de semantiek van de data als beste snapt, een grote rol heeft in het algoritme. Zie 3.1 voor meer informatie over deeldimensies.

```
foreach (DimTbls as DimTbl){
        if (Usr_Inputs_ExtraDim(DimTbl)){
                createExtraColumns(dimTbl, {columns});
                executeExplodeAlgorithm("algorithm_name", dimTbl);
        }
}
```

   De vereisten voor dit algoritme is dat de gebruiker interactief het algoritme stuurt en dat het transformatiealgoritme naar de deeldimensies bekend is.

   Er is besloten om een bibliotheek van transformaties aan te leggen die een aantal basis transformaties kan uitvoeren. Bij ieder nieuw voorkomen, voeg je een algoritme toe. Op deze manier garandeer je continuïteit binnen je *WOLAP* applicatie.

Stap 2a

Voor het samenstellen van het additioneel datamodel dienen we alleen de profilinginterpreter een aantal tabellen voor te laten berekenen.

Dit klinkt simpeler dan het in praktijk zal zijn. Je kunt namelijk selecteren binnen of buiten groepen. Aangezien we aan group-sharing gaan doen lijkt de keuze eenvoudig, namelijk binnen een groep. Hierdoor verliezen we alleen het overkoepelende idee van group-sharing uit het oog. Er zal ook een keuze gemaakt moeten worden naar de totalen, om goed af te wegen welke dimensies je gaat voorberekenen.

Laten we gaan kijken naar een algoritme.

Zij *selectMaxN ()* een functie die van alle profielen de N meest gevraagde dimensies oplevert, plus extra informatie over de grootte van de groep en de diversiteit in gebruikers

Zij *removeDoubles ()* een functie die de dubbele dimensies verwijdert en de dimensie informatie die overblijft update, met de grootte van de groep en de diversiteit in gebruikers.

```
for (i=0; i < x; i++) {
        max[] = selectMaxN(i);
}
max = removeDoubles(max);
```

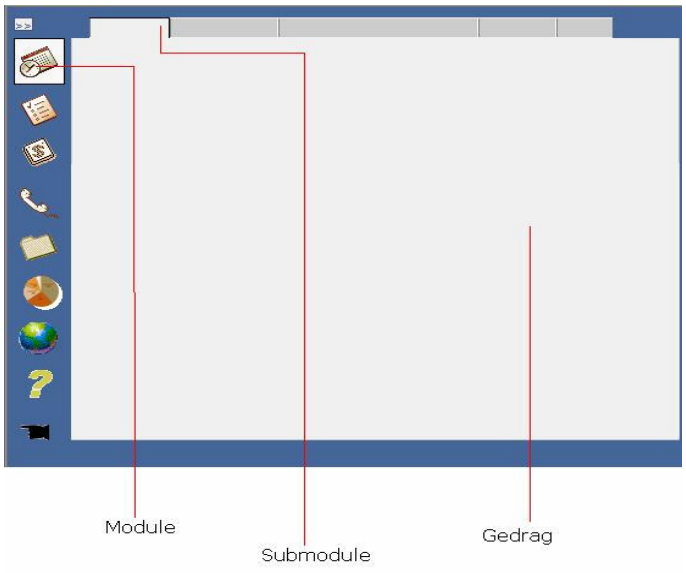
Je zou nu kunnen stellen dat het genereren van al deze dimensies een basis geeft. Er moet echter een procedure zijn die deze tabellen bijhoudt. Bij iedere vraag van een gebruiker kan het immers veranderen. Er is besloten dit empirisch te gaan onderzoeken tijdens de implementatiefase.

Het is zeker interessant om te bepalen hoe een doorsnijding gegenereerd wordt. Sla je hierin alleen de getallen op, of ook de functionaliteit die achter de getallen komt. Het antwoord voor onze doeleinden is eenvoudig. De functionaliteit zal ook opgeslagen worden om ervoor te zorgen dat de gebruiker alleen dient te wachten op het bepalen of het gegenereerde data betreft of nieuw gecalculeerde data. Daarnaast hoeft de gebruiker alleen te wachten op de datatransportatie naar de client. De precieze uitwerking zal wederom empirisch worden bepaald tijdens het implementeren.

Stap 3
Voor het aanbieden van het model als zijnde dataset is een gemakkelijke stap als de *WOLAP* tool geschikt gemaakt wordt voor dit model. Dit klinkt flauw, maar het echte werk zit niet in de *WOLAP* tool, want dat wordt geschreven naar aanleiding van het model. Het aanbieden zelf geschied op basis van het Application Framework van Efficiency Online.

Dit Framework werkt globaal als volgt[6] (Figuur 5.1.2):

1. Iedere tabel is een object. Een aantal methoden van dit object representeren de kolommen.
2. Ieder object erft standaard functionaliteit van een Base object. Dit om algemene methoden toegankelijk te maken op ieder willekeurig object.
3. Relaties worden expliciet gebruikt. Deze moeten in een datamodel volgens een syntax afspraak worden opgenomen. Vervolgens worden deze automatisch gedetecteerd en beschikbaar gemaakt in het systeem. Dit werkt zowel direct als gerelateerd.
4. Om een functionaliteit beschikbaar te maken in het Framework voeg je deze toe als een zijnde een gedrag. *WOLAP* zal een dergelijk gedrag zijn, evenals de transformatie methodiek.
5. Ieder gedrag kan gebruik maken van alle objecten en standaard functionaliteit die wordt aangeboden door het Application Framework. Denk aan een Database abstractielaag een SQL object (deze genereert queries voor een database), een grafische interface, een bibliotheek aan triviale en complexe functies, etc.



Figuur 5.1.2 *Application Framework*

[6] Deze beschrijving zal summier gehouden worden om niet teveel informatie openbaar te maken. De specifieke werking is voor deze scriptie niet van belang. Voor het daadwerkelijke implementeren wel, maar dit zal geschieden na het afstuderen.

5.2 Heuristiek

In deze paragraaf worden standaard heuristieken bepaald en zal er ook beschreven worden hoe gedetecteerd kan worden of er deeldimensies geëxtraheerd kunnen worden van een brondimensie.

Eerst vatten we samen welke informatie we altijd hebben:

- het aantal rijen:
  Per cel:
  - kolomnaam
  - waarde
  - uniciteit van deze waarde in relatie tot alle waarden in deze kolom van de gehele tabel

- het aantal kolommen:
  Per kolom:
  - naam
  - type
  - lengte
  - NULL (mag een waarde in deze kolom expliciet leeg zijn of niet)
  - database afhankelijke relaties / functies
  - Deze zijn in het Application Framework aanwezig door middel van syntax. Een foreign key wordt gerepresenteerd door <*kolomnaam*>_id of als er meerdere foreign keys bestaan binnen een tabel <*kolomnaam*>_l_<*label*>_id.

Een dimensie wordt getransformeerd naar een foreign key als *(of)*:

- het type van een kolom geen getal is. We bekijken hier de reële getallen.
- het getal, qua aantal karakters, lang en als de uniciteit klein genoeg is, om deze kolom te transformeren.

Een dimensie wordt opgesplitst in deeldimensies als *(of)*:

- het type van een kolom een datum is
- de interpreter voor deeldimensies de waarden van een kolom herkent als zijnde een dimensie die deelbaar is, vanuit zijn bibliotheek.
- de beheerder van de dataset aangeeft dat een verdeling naar deeldimensies mogelijk is.

Een dimensie wordt uit de dataset verwijderd als *(of)*:

- de dimensie een deeldimensie vormt van een vaderdimensie.
- de dimensie irrelevant wordt geacht voor verdere verwerking

6. Implementatie

In dit hoofdstuk zal beschreven worden hoe het algoritme, dat in hoofdstuk 5 beschreven werd, geïmplementeerd is. Allereerst zal beschreven worden welke programmatuur is gebruikt en hoe resultaten berekend worden. De broncode is bijgevoegd in bijlage A.

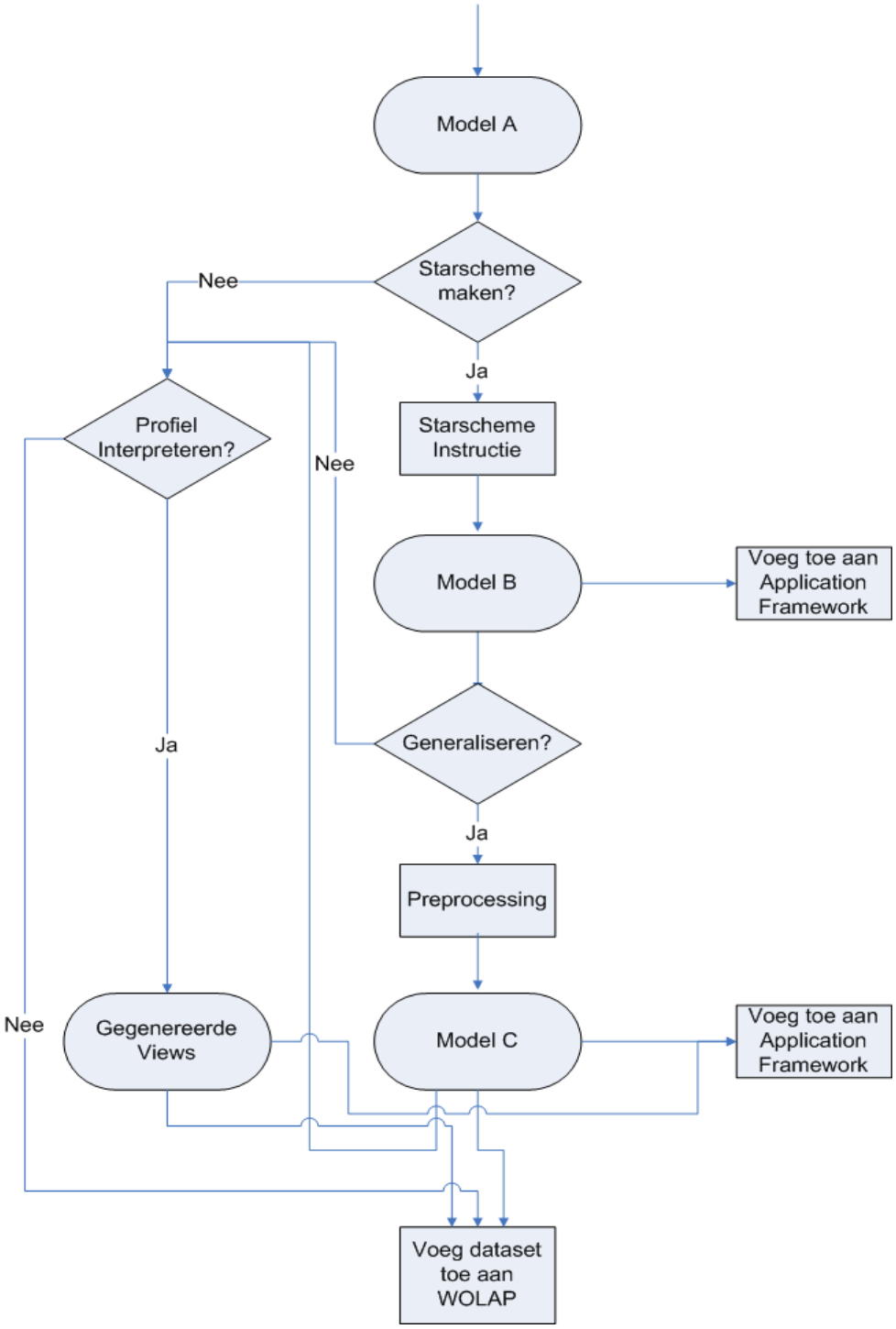
6.1 Inleiding

De gekozen taal is PHP, de gekozen database is MySQL. Deze keuze komt voort uit de gebruikte programmeertaal en database bij Efficiency Online. Aangezien het Application Framework geschreven is in PHP leek dit qua programmeertaal het makkelijkst aansluiten.
Het Application Framework gebruikt een database abstractie laag en bevat een query object[7]. Deze zorgen er tezamen voor dat het Application Framework database onafhankelijk is. De keuze voor MySQL als database is puur en alleen op eigen ervaring gebaseerd. MySQL is gratis en vrij snel. Helaas is MySQL niet zover in het afdwingen van constraints en andere belangrijke eigenschappen die concurrenten wel bezitten. Vanwege de snelheid en mijn werkervaring met deze database server is gekozen voor MySQL.

De gebruikte programmeermethode probeert een object georiënteerde benadering te nemen. Aangezien PHP niet echt een object georiënteerde taal is zal dit met een korreltje zout genomen moeten worden.

In figuur 6.1 is uitgewerkt hoe de datamanager het transformatieproces zal uitvoeren. Dit is het belangrijkste component. De datamanager zal compleet in het systeem geschreven worden. Hiervoor maak ik gebruik van het "Wizardbehaviour[8]" dat ik met Daniel van der Wallen voor Efficiency Online zal ontwikkelen. De keuzemogelijkheden zullen geconfigureerd kunnen worden zodat de aanpasbaarheid van de datamanager niet inlevert voor functionaliteit. Voor deze stroom is gekozen om de beheerder te laten kiezen hoe hij de data in een *WOLAP* applicatie wil hebben. Voor een kleine dataset ga je niet het hele transformatieproces doorlopen, dit zal alleen gebeuren bij middel tot grote datasets. De verantwoordelijkheid van de keuze leggen we hierdoor bij de beheerder. Er zal echter geprobeerd worden zoveel mogelijk feedback te leveren over de automatisch gemaakte keuzes, zodat een beheerder te allen tijde kan voorspellen of een nieuw te maken model een vooruitgang is.

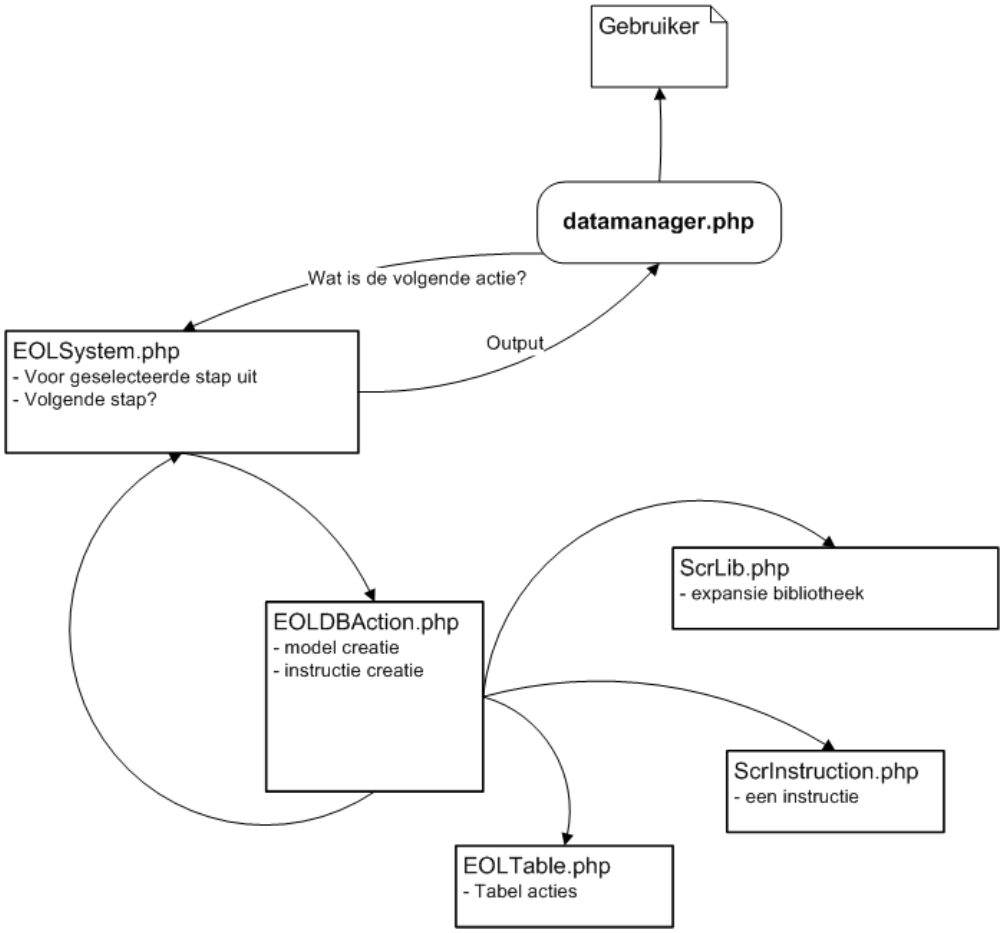
[7] Een object dat een interface bevat voor het opbouwen van queries.
[8] Een behaviour (gedrag) wordt in het Application Framework geladen in een submodule.



Figuur 6.1 *Implementatie model*

6.2 Objectbeschrijving *Datamanager*

In deze paragraaf zal het objectmodel van de datamanager, de transformatie tool, gegeven worden. Hierdoor is inzicht te verkrijgen in de werking van de tool en is de implementatie aan de box uit hoofdstuk 5 te koppelen. Het is een vraaggestuurd systeem, deze vraag wordt aan de systeemklasse gesteld en deze handelt tevens het huidige antwoord af.



Figuur 6.2.1 *Objectbeschrijving van de datamanager*

Zoals te zien is in figuur 6.2.1 handelt EOLDBAction.php alle belangrijke stappen af. Alle basis acties voor het opbouwen van database acties zijn uitbesteedt aan EOLTable.php. Voor verdere details over de methoden per klasse, verwijs ik naar bijlage A.

6.3 Proof of concept *WOLAP* tool

Om een conclusie te kunnen trekken van de werking van de box, is ervoor gekozen een proof of concept *WOLAP* tool te bouwen. Dit is een test applicatie. Er wordt per gerelateerd item een basis query gegenereerd op basis van de gecreëerde instructieset. Deze query bevat een simpele doorsnijding. De gegenereerde queries worden uitgevoerd op ieder model dat op dat moment beschikbaar is. De resultaten, executietijd per dimensie, worden in een grafiek uitgezet. Hieruit kan men afleiden wat een standaard query voor performance verbetering oplevert per model. De queries die gegenereerd worden zijn standaard aangezien er dan gemakkelijk uitspraken gedaan kunnen worden over de gemiddelde performance.

Aangezien het flink wat tijd kost om te analyseren welke query nu het zwaarst gaat zijn voor een dataset, zullen deze extremen niet verwerkt worden in de conclusie over een model. Tevens zal het beeld van de modellen dan scheef zijn. Het is gemakkelijk te beredeneren, dat als een standaard query sneller op model B draait dan op model A, dit ook het geval zal zijn met zwaardere queries.

Het omgekeerde geval vormt echter een probleem. Als model A sneller / even snel is dan model B, dan wil dit niet zeggen dat dit ook geldt voor de zwaardere queries. Is er in dat geval gekozen voor het gerelateerd maken van een kolom, dan zal het verschil in performance procentueel berekend worden om het verschil aan te geven. Daarnaast zal met behulp van de kosten (in opslagruimte) van de interne representatie van de datatypes in MySQL voorspeld kunnen worden wat het gedrag zal gaan zijn. Dit zal echter nu door de beheerder gedaan moeten worden omdat het teveel werk behelst een compleet wetenschappelijk verantwoorde test applicatie te schrijven. Vandaar dat het deel dat twijfelachtig is in de huidige test applicatie afgevangen kan worden met redenatie. Zie tabel 6.3.1 voor de kosten (in opslagruimte) van de interne representatie van de datatypes in MySQL.

Kolom Type	Benodigde Ruimte
TINYINT	1 byte
SMALLINT	2 bytes
MEDIUMINT	3 bytes
INT, INTEGER	4 bytes
BIGINT	8 bytes
FLOAT(*p*)	4 bytes | 0 <= *p* <= 24, 8 bytes | 25 <= *p* <= 53
FLOAT	4 bytes
DOUBLE [PRECISION], item REAL	8 bytes
DECIMAL(*M*,*D*), NUMERIC(*M*,*D*)	*M*+2 bytes | *D* > 0, *M*+1 bytes | *D* = 0 (*D*+2, if *M* < *D*)
DATE	3 bytes
DATETIME	8 bytes
TIMESTAMP	4 bytes
TIME	3 bytes
YEAR	1 byte
CHAR(*M*)	*M* bytes, 0 <= *M* <= 255
VARCHAR(*M*)	*L*+1 bytes, | *L* <= *M* and 0 <= *M* <= 255
TINYBLOB, TINYTEXT	*L*+1 bytes, | *L* < 2^8
BLOB, TEXT	*L*+2 bytes, | *L* < 2^16
MEDIUMBLOB, MEDIUMTEXT	*L*+3 bytes, | *L* < 2^24
LONGBLOB, LONGTEXT	*L*+4 bytes, | *L* < 2^32
ENUM('*value1*','*value2*',...)	1 of 2 bytes, afhankelijk van het aantal enumeratie items (maximaal 65,535 items)
SET('*value1*','*value2*',...)	1, 2, 3, 4, of 8 bytes, afhankelijk van het aantal items in de verzameling (maximaal 64 items)

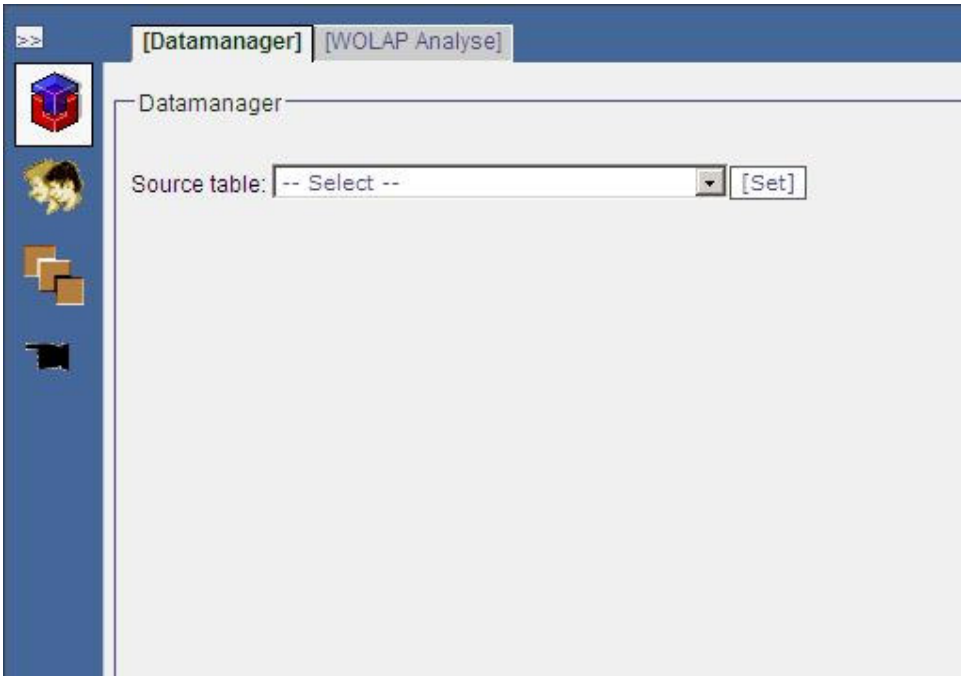
Figuur 6.3.1 *Kosten (in opslagruimte) van de interne representatie van de datatypes in MySQL*

6.4 De datamanager

In deze paragraaf zal stap voor stap de datamanager besproken worden. Hierdoor is te zien hoe de applicatie voor de gebruiker zal werken.

Stap 1 (figuur 6.4.1):

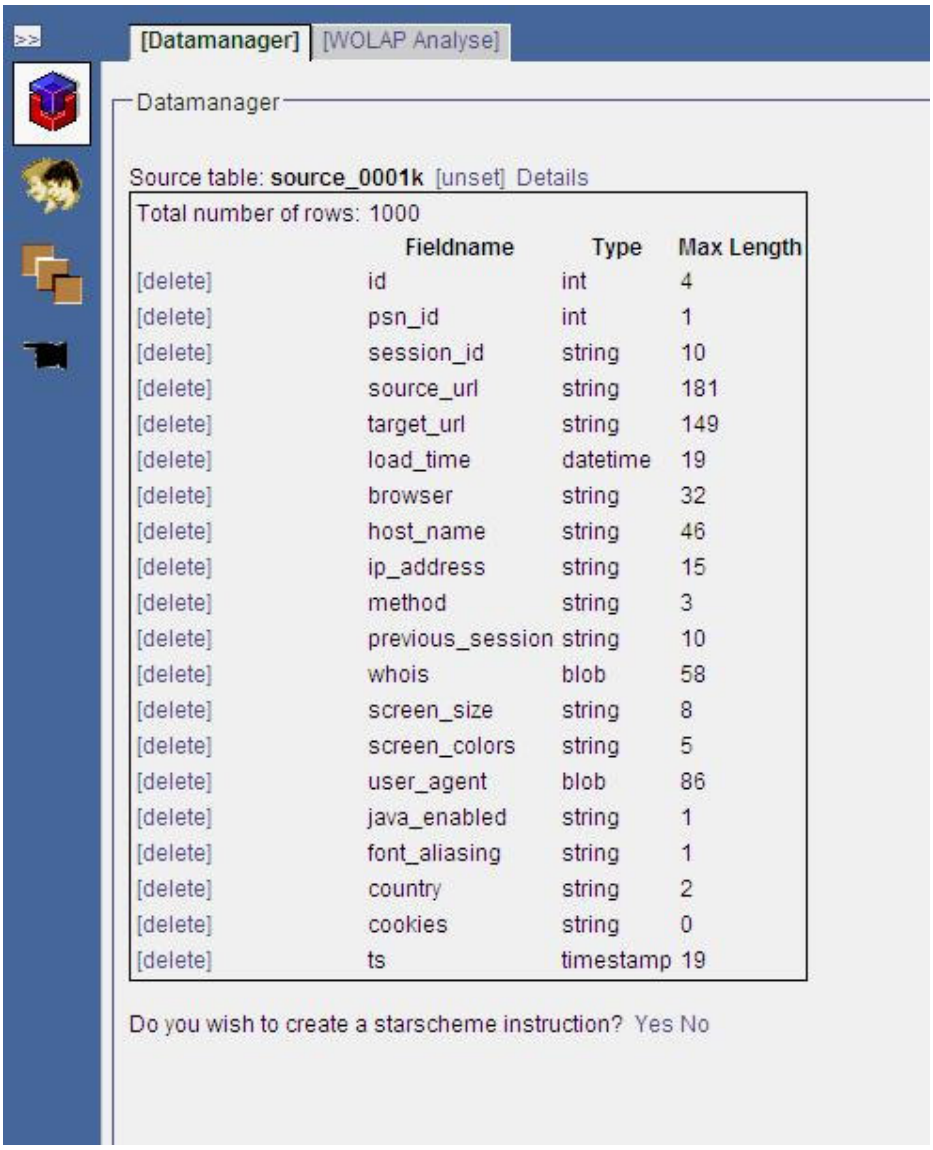
In deze stap zal de gebruiker een tabel (de brontabel) kiezen waarmee deze het stappenplan zal gaan doorlopen. De gebruiker krijgt de mogelijkheid te kiezen uit alle tabellen die beschikbaar zijn binnen de applicatiedatabase.



Figuur 6.4.1 *Stap 1, selecteer een brontabel, model A*

Stap 2 (figuur 6.4.2):

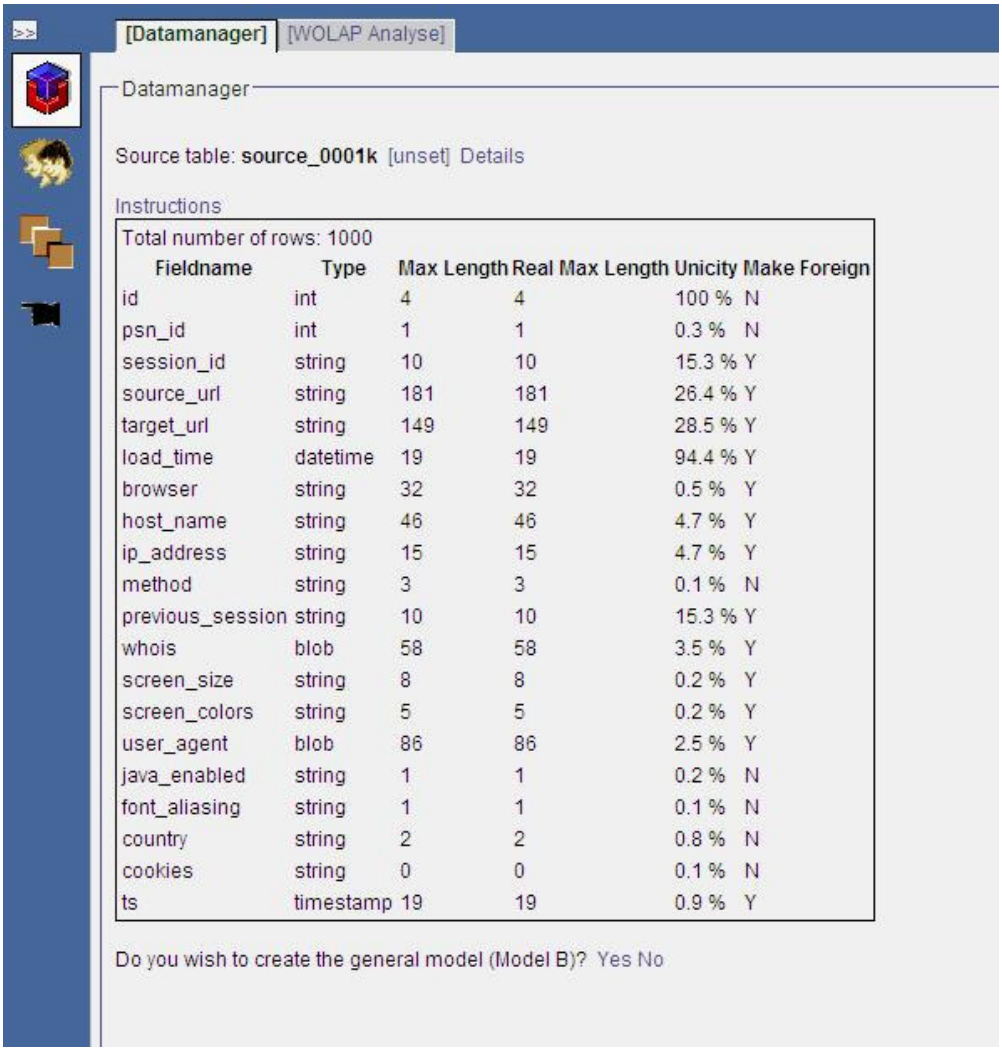
In deze stap kan de gebruiker kiezen of deze een instructieset voor de transformaties wil laten genereren. Hij kan tevens zien welke velden beschikbaar zijn binnen de tabel en eventueel velden verwijderen die hij niet nodig acht.



Figuur 6.4.2 *Stap 2, de tabelinformatie en instructie creëer vraag.*

Stap 3 (figuur 6.4.3):

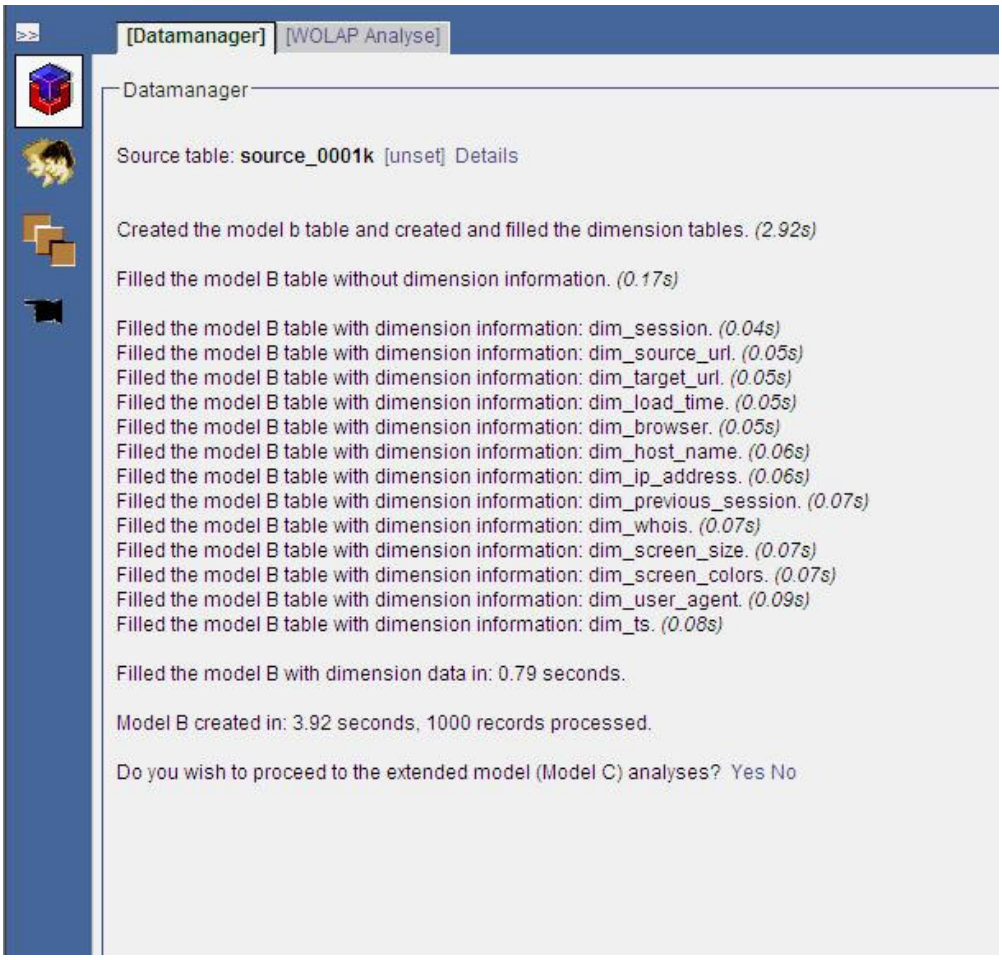
In deze stap kan de gebruiker de gegenereerde instructieset bekijken. Hierdoor kan deze inschatten of het maken van model B nodig is. Tevens krijgt deze de vraag gesteld of er verder gegaan moet worden met het creëren van model B.



Figuur 6.4.3 *Stap 3, de instructieset en model B creëer vraag.*

Stap 4 (figuur 6.4.4):

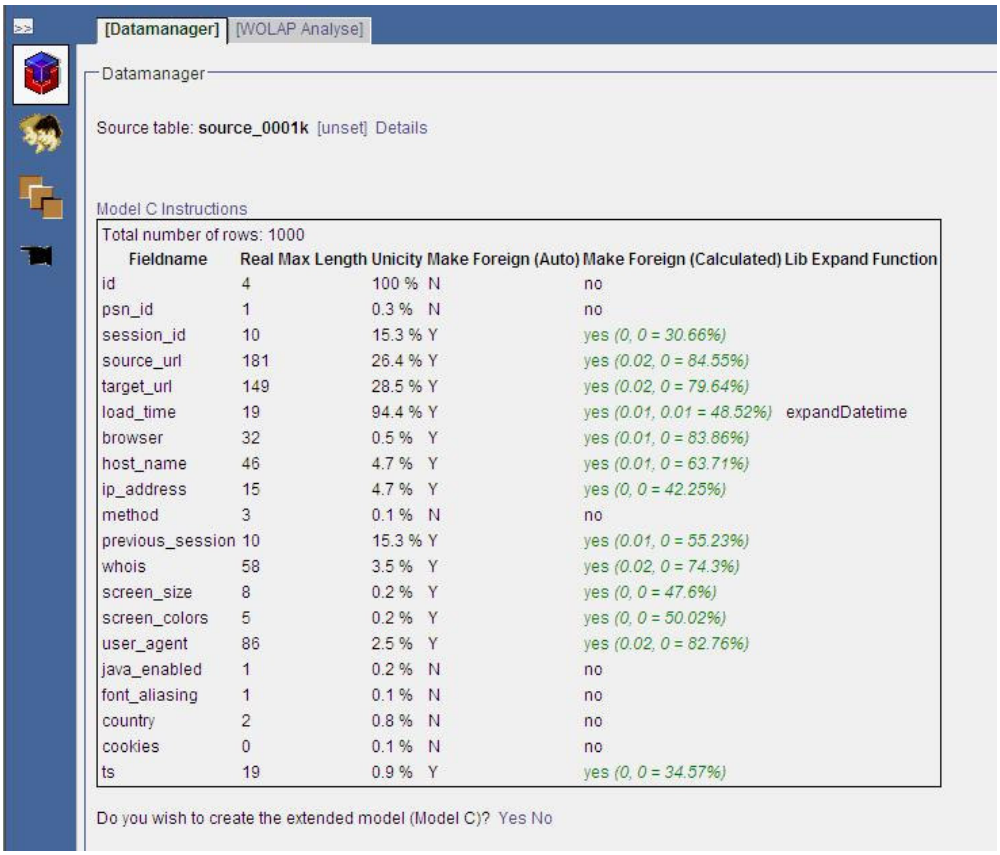
In deze stap kan de gebruiker de gegevens van het maken van model B bekijken. Hieruit kan volgen welke dimensies duur zijn. Tevens krijgt de gebruiker de vraag gesteld of deze model B wil analyseren voor het maken van model C.



Figuur 6.4.3 *Stap 3, een samenvatting van het maken van model B en model C analyse creëer vraag.*

Stap 5 (figuur 6.4.5):

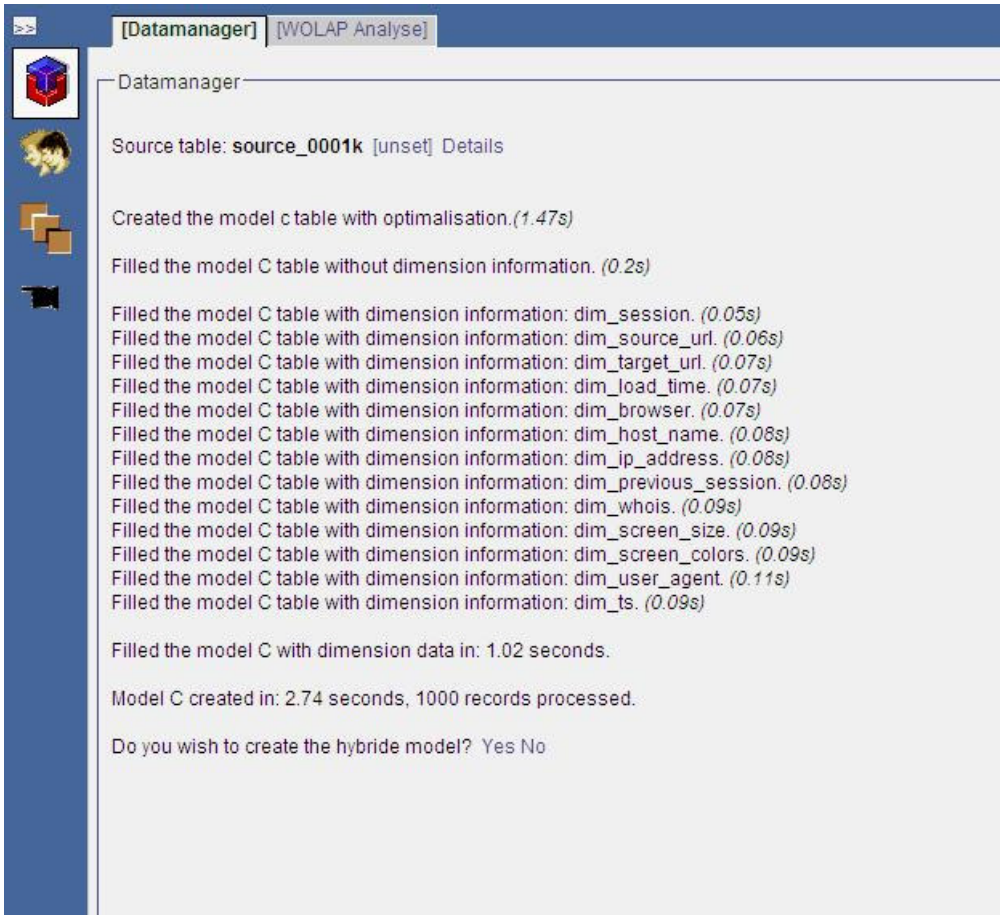
In deze stap kan de gebruiker zien hoe model B het doet tegenover model C, of de stappen om het gerelateerd te maken gunstig zijn (procentueel gezien) en of er een kolom geëxpandeerd kan worden. Tevens kan de gebruiker kiezen om model C te creëren.



Figuur 6.4.5 *Stap 5, een samenvatting van het maken van de model B analyse en model C creëer vraag.*

Stap 6 (figuur 6.4.6):

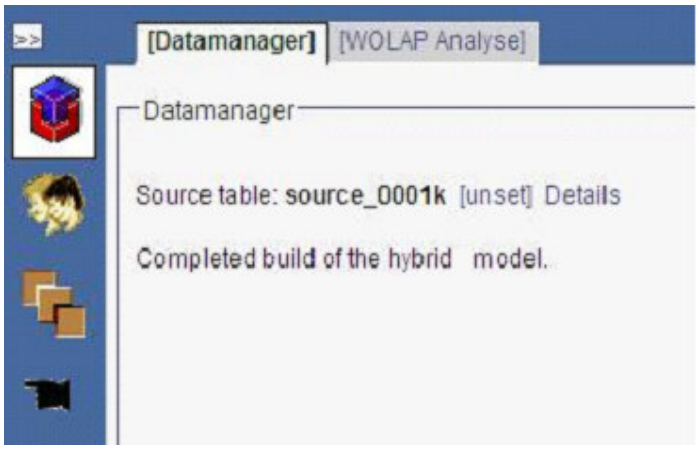
In deze stap kan de gebruiker de gegevens van het maken van model C bekijken. Tevens krijgt de gebruiker de vraag gesteld of deze het hybride model wil maken, de laatste stap in het proces.



Figuur 6.4.6 *Stap 6, een samenvatting van het maken van model C en hybride model creëer vraag.*

Stap 7 (figuur 6.4.7):

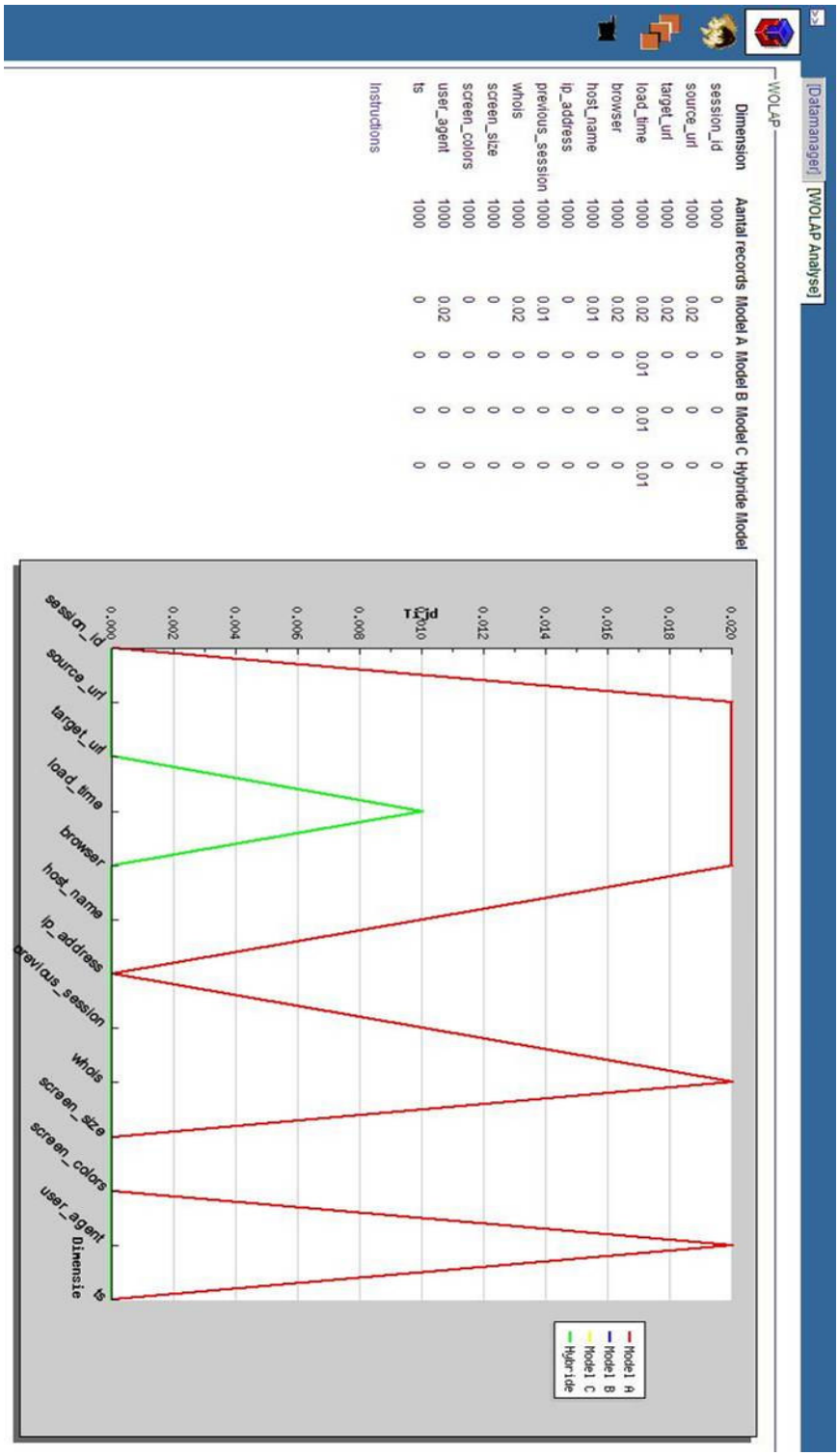
In deze stap krijgt de gebruiker het bericht dat alle stappen van het proces doorlopen zijn. Het complete, samengestelde model is af.



Figuur 6.4.7 *Stap 7, het proces compleet doorlopen.*

De analyse (figuur 6.4.8):

Om uitspraken te kunnen doen over de gemaakte stappen kan de gebruiker nu de WOLAP analyse uitvoeren. De resultaten geven weer wat er bereikt is met het transformeren van de data, voor WOLAP doeleinden.



Figuur 6.4.8 *De resultaten van de analyse uitgezet in een grafiek.*

6.5 Testen

De testsets die zijn gebruikt worden beschreven in paragraaf 6.5.1 en de testresultaten in paragraaf 6.5.2.

6.5.1 Testsets

In deze paragraaf zal beschreven worden welke testsets er zijn gebruikt en hoe deze zijn opgebouwd. Er is gekozen voor een set van aanroepen van webpagina's.

Originele set: 250 records
Tabeldefinitie:

1. id *(int 11)*
2. psn_id *(int 11)*
3. session_id *(varchar 10)*
4. source_url *(varchar 255)*
5. target_url *(varchar 249)*
6. load_time *(datetime)*
7. browser *(varchar 200)*
8. host_name *(varchar 100)*
9. ip_address *(varchar 25)*
10. method *(varchar 10)*
11. previous_session *(varchar 50)*
12. whois *(text)*
13. screen_size *(varchar 20)*
14. screen_colors *(varchar 20)*
15. user_agent *(text)*
16. java_enabled *(char 1)*
17. font_aliasing *(char 1)*
18. country *(char 2)*
19. cookies *(char 1)*
20. ts *(timestamp)*

Alle testsets zijn hiervan afgeleid en zijn een subset van of een aantal maal deze originele set.
Uit de tabeldefinitie is duidelijk af te lezen dat alle veelvoorkomende kolomtypes voorkomen. De lengte van de kolommen is reeds geoptimaliseerd om geen valse resultaten te krijgen voor geheugen gebruik en benodigde opslagruimte. In de praktijk zullen de resultaten hierdoor nog iets positiever uitpakken, maar dit verschil is te verwaarlozen en niet interessant voor uitspraken over het ontwikkelde model.

De testsets:

Naam	Aantal Records	% van originele testset	Gebruikt voor:
source_0001k	1.000	0.4	Ontwikkeling
source_0050k	50.000	20	Normale grootte test
source_0100k	100.000	40	Normale grootte test
source_0250k	250.000	100 ***(origineel)***	Grote grootte test
source_0500k	500.000	200 *(2x het origineel)*	Grote grootte test
source_1000k[9]	1.000.000	400 *(4x het origineel)*	Enorme grootte test

Er is gekozen voor deze dataset, omdat de inhoud overeenkomt met het werk dat Efficiency Online doet. Daarnaast is de verscheidenheid van datatypes een goede manier om het bepalen van het gerelateerd maken van de verschillende kolommen te testen en te beredeneren.

[9] Hiervan is als aardigheid alleen model B gecreëerd.

6.5.2 Testresultaten

De resultaten zijn per testset uitgezet tegen ieder model. Er is gekozen om de dataset met 100.000 records uitgebreid te bespreken. Van de overige datasets zijn de resultaten uiteraard wel opgenomen. Er is gekozen voor deze dataset omdat is gebleken dat deze dataset het meest representatief is voor grootte van tabellen bij Efficiency Online.

Allereerst wat basis gegevens over het maken van de verschillende modellen per dataset:

Dataset	# Records	Model B (in sec)	Model C (in sec)
source_0001k	1.000	3.92	2.74
source_0050k	50.000	62.23 (+/- 1 min.)	83.48 (+/- 1.5 min.)
source_0100k	100.000	122.52 (+/- 2 min.)	172.93 (+/- 3 min.)
source_0250k	250.000	319.55 (+/- 5.5 min.)	472.51 (+/- 8 min.)
source_0500k	500.000	657.32 (+/- 11 min.)	913.54 (+/- 15.5 min.)
Source_1000k	1.000.000	+/- 33 minuten	-

Details source_0100k:

De tabeldefinitie is gelijk aan degene die besproken is in paragraaf 6.5.1. Uit de basisgegevens is af te lezen dat het maken van een compleet model niet langer dan tien minuten zal duren voor het doorlopen en bekijken van het gehele proces. Het is nu interessant te kijken naar de resultaten van de *WOLAP* analyse.

De resultatentabel is als volgt opgebouwd:

- Dimension:
  de kolomnaam

- Aantal records:
  het aantal records van de dataset

- Model A:
  het aantal seconden dat het duurt om met de kolom de standaard WOLAP query te executeren op model A. (Afgerond op 2 decimalen)

- Model B:
  het aantal seconden dat het duurt om met de kolom de standaard WOLAP query te executeren op model B. (Afgerond op 2 decimalen)

- Model C:
  het aantal seconden dat het duurt om met de kolom de standaard WOLAP query te executeren op model C. (Afgerond op 2 decimalen)

- Hybride model:
  het aantal seconden dat het duurt om met de kolom de standaard WOLAP query te executeren op het hybride model. (Afgerond op 2 decimalen)

Resultatentabel van source_0100k

Dimension	Aantal records	Model A	Model B	Model C	Hybride Model
session_id	100000	2.19	0.81	0.50	0.00
source_url	100000	2.10	0.21	0.22	0.00
target_url	100000	2.06	0.20	0.22	0.00
load_time	100000	0.99	0.68	0.70	0.70
browser	100000	1.30	0.19	0.19	0.20
host_name	100000	3.00	0.38	0.37	0.37
ip_address	100000	0.68	0.38	0.38	0.37
previous_session	100000	1.98	0.48	0.48	0.47
whois	100000	1.89	0.21	0.21	0.21
screen_size	100000	0.39	0.21	0.21	0.21
screen_colors	100000	0.38	0.19	0.19	0.19
user_agent	100000	2.00	0.21	0.21	0.21
ts	100000	0.28	0.18	0.17	0.17

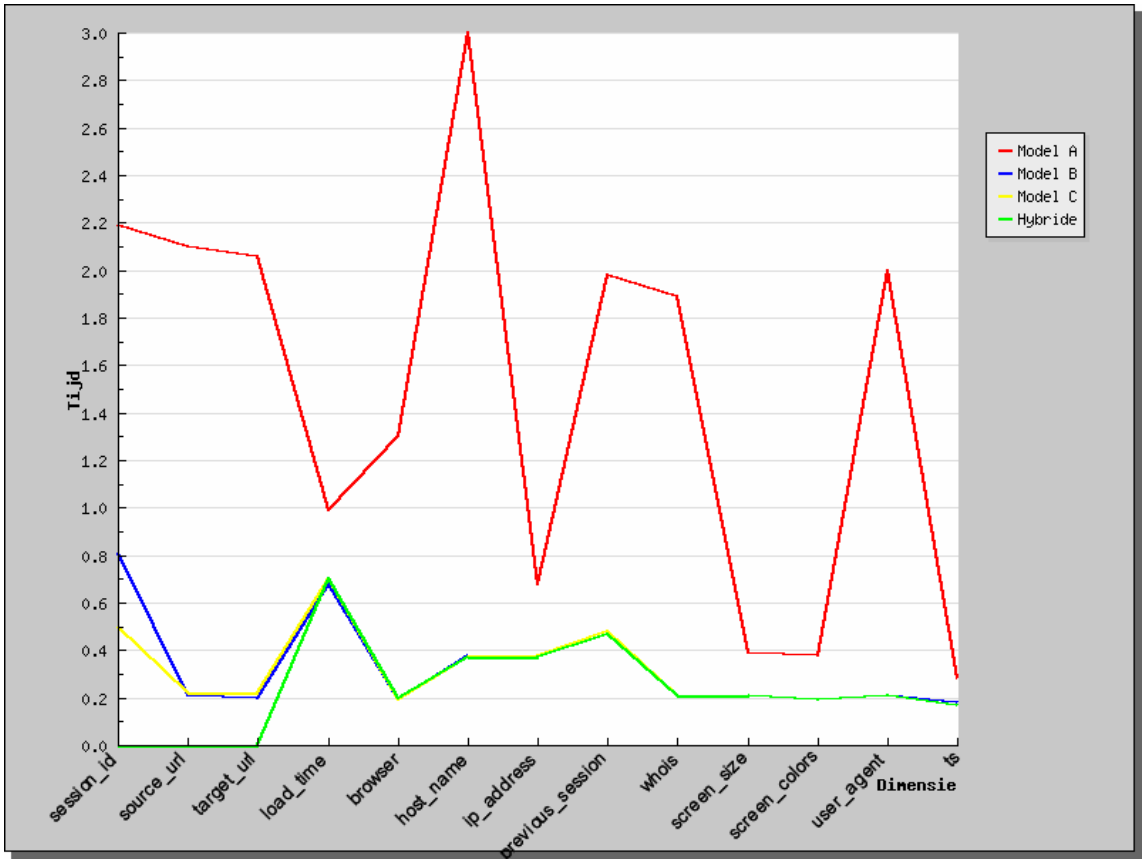
0 : de voorgegenereerde views.

Als er in het hybride model geen voorkomen in de voorgegenereerde view beschikbaar is zal de query uitgevoerd worden op model C. Hiervoor is gekozen omdat van model C gezegd kan worden dat deze het rijkst is, qua gegevens en in de meeste gevallen ook snelheid.

In figuur 6.5.2.1 is bovenstaande tabel grafisch weergegeven.

De figuur dient als volgt gelezen te worden en is opgebouwd met de volgende regels:

1. Er zijn geen trends te ontdekken ten opzichte van de verschillende dimensies, er dient gekeken te worden naar de tijd per dimensie
2. Door de verbinding te lezen als verspringing, kan per model gekeken worden of de curve ook zo stijl is.
3. Door te kijken naar de instructieset, figuur 6.5.2.2, kan men met behulp van de uniciteit, het kolomtype en de lengte van de voorkomens in de kolom verklaren waarom de standaard query duur of goedkoop is.
4. Alleen de gerelateerd gemaakt items doen mee in de analyse. Dit om eventuele kleine winst van grootte weg te snijden. Er wordt aangenomen dat de kolommen die met rust zijn gelaten tijdens de transformatie in ieder model dezelfde performance hebben.



Figuur 6.5.2.1 *WOLAP Analyse output. (Gecreëerd door de WOLAP Analyse tool met behulp van JpGraph, http://www.aditus.nu/jpgraph/index.php)*

Conclusies die te trekken zijn uit bovenstaande grafiek zijn bijvoorbeeld:

1. Het creëren van het model heeft wel degelijk zin
2. load_time is een dimensie die erg duur is in het geoptimaliseerde model en qua performance niet echt veel beter dan in model A.
3. voor de dimensies screen_size, screen_colors en ts is het niet nodig om deze gerelateerd te maken. De winst in tijd is te klein, al is deze in percentage wel groot genoeg.
4. session_id is uiterst geschikt om voor te genereren, evenals load_time en previous_session. source_url daarentegen is minder geschikt, maar wordt wel vaak gebruikt (kennis van de profiling)
5. De algehele performance is gelijk getrokken, het wachten zal overal ongeveer even lang duren. Zou je punt 4 meenemen zal dit de vlakheid bevorderen met alleen uitschieters naar beneden in tijd, dit werkt prettiger dan andersom.

De bevindingen zullen verwerkt moeten worden in de datamanager. Om dit te kunnen gaan doen zal het noodzakelijk zijn de instructieset die gemaakt is opnieuw te kunnen inladen. Op die manier kan een model handmatig geoptimaliseerd worden. Dit is een goed idee voor de toekomst.

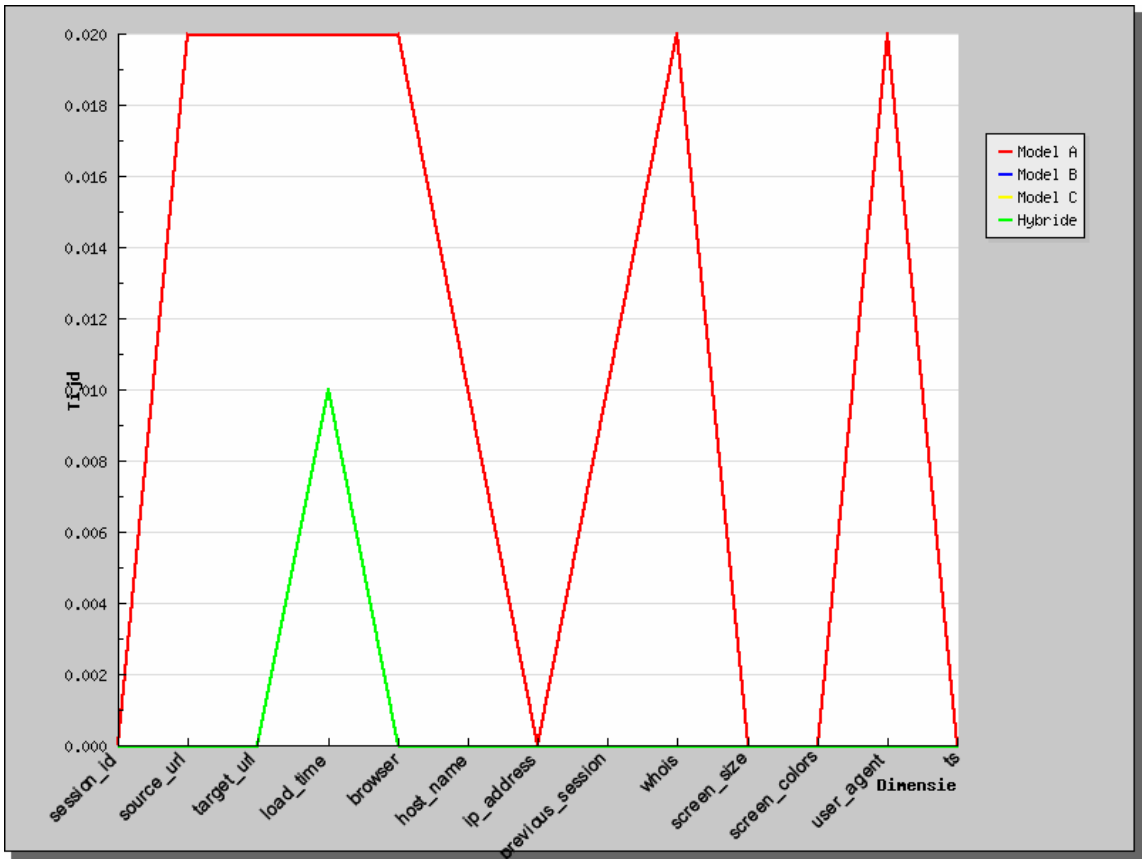
Fieldname	Type	Max Length	Real Max Length	Unicity	Make Foreign
id	int	6	6	100.00 %	N
psn_id	int	1	1	0.01 %	N
session_id	string	10	10	55.85 %	Y
source_url	string	255	255	6.37 %	Y
target_url	string	249	249	4.30 %	Y
load_time	datetime	19	19	97.67 %	Y
browser	string	34	34	0.03 %	Y
host_name	string	50	50	33.64 %	Y
ip_address	string	15	15	33.16 %	Y
method	string	3	3	0.00 %	N
previous_session	string	10	10	49.24 %	Y
whois	blob	75	75	4.00 %	Y
screen_size	string	9	9	0.13 %	Y
screen_colors	string	10	10	0.02 %	Y
user_agent	blob	139	139	3.41 %	Y
java_enabled	string	1	1	0.00 %	N
font_aliasing	string	1	1	0.00 %	N
country	string	2	2	0.17 %	N
cookies	string	1	1	0.00 %	N
ts	timestamp	19	19	0.35 %	Y

Figuur 6.5.2.2 *Instructieset van source_0100k*

Met behulp van de instructieset is het mogelijk de uiteindelijke performance te optimaliseren en de huidige gegevens te analyseren.

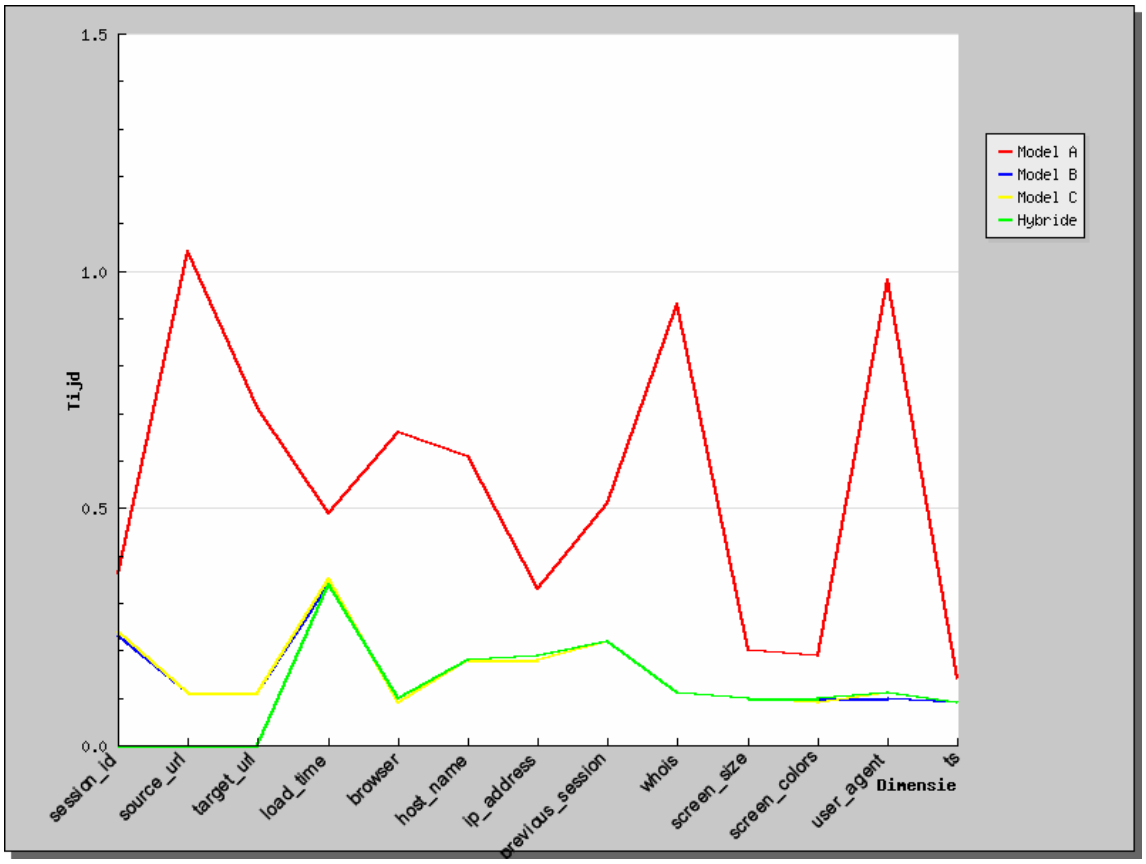
Overige tabellen en de bijbehorende *WOLAP* Analyse:

source_0001k:



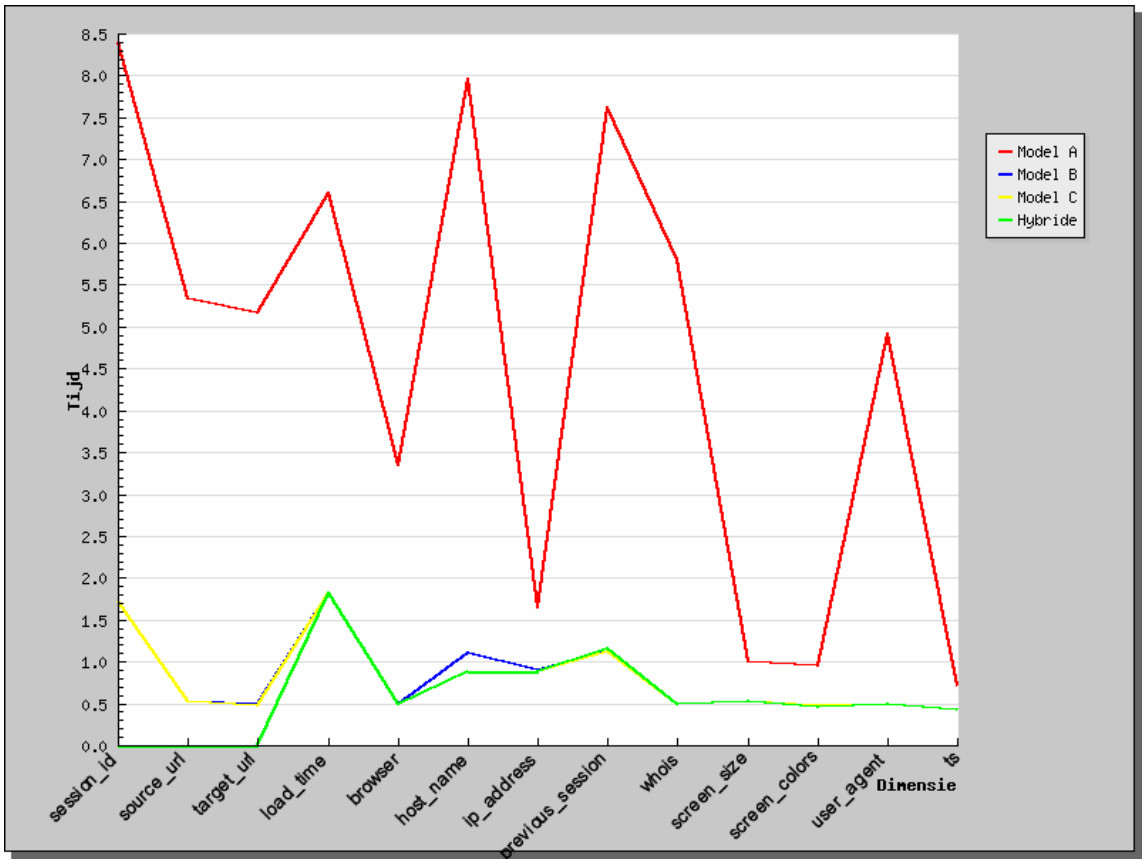
Figuur 6.5.2.3 *WOLAP Analyse van source_0001k*

source_0050k:



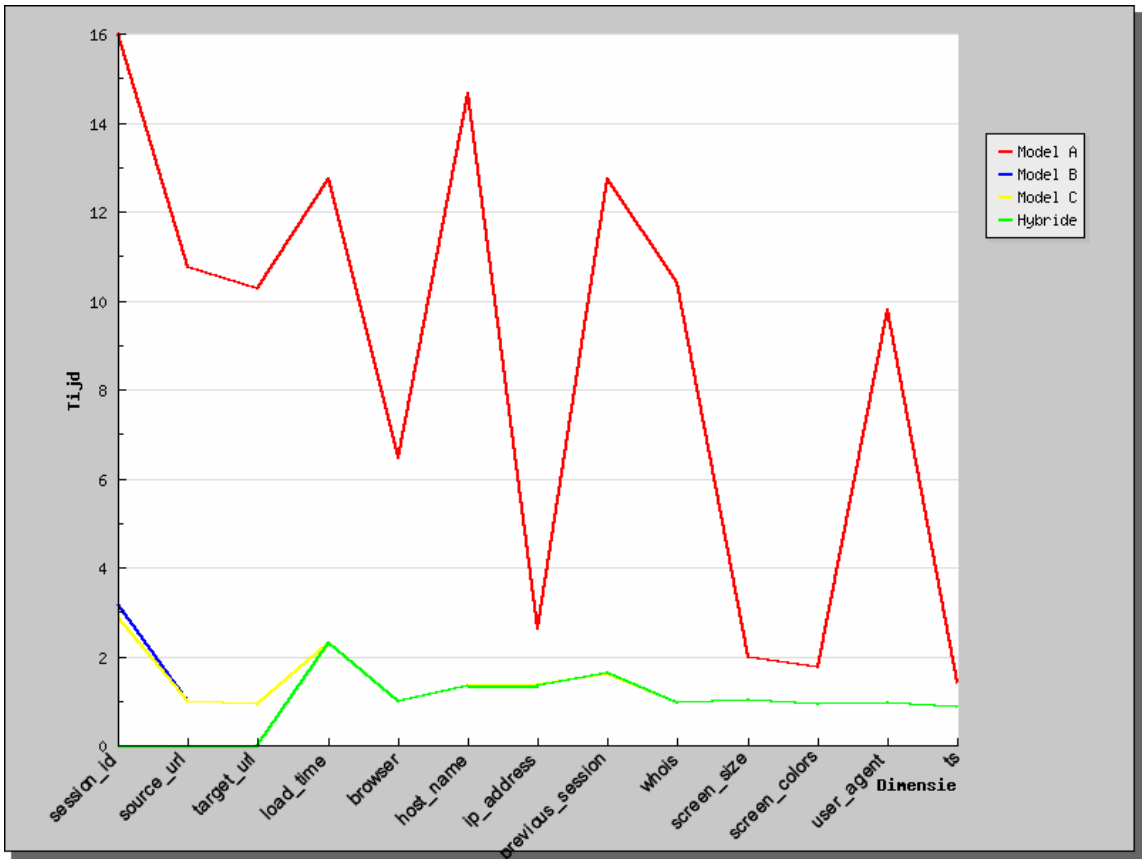
Figuur 6.5.2.4 *WOLAP Analyse van source_0050k*

source_0250k:



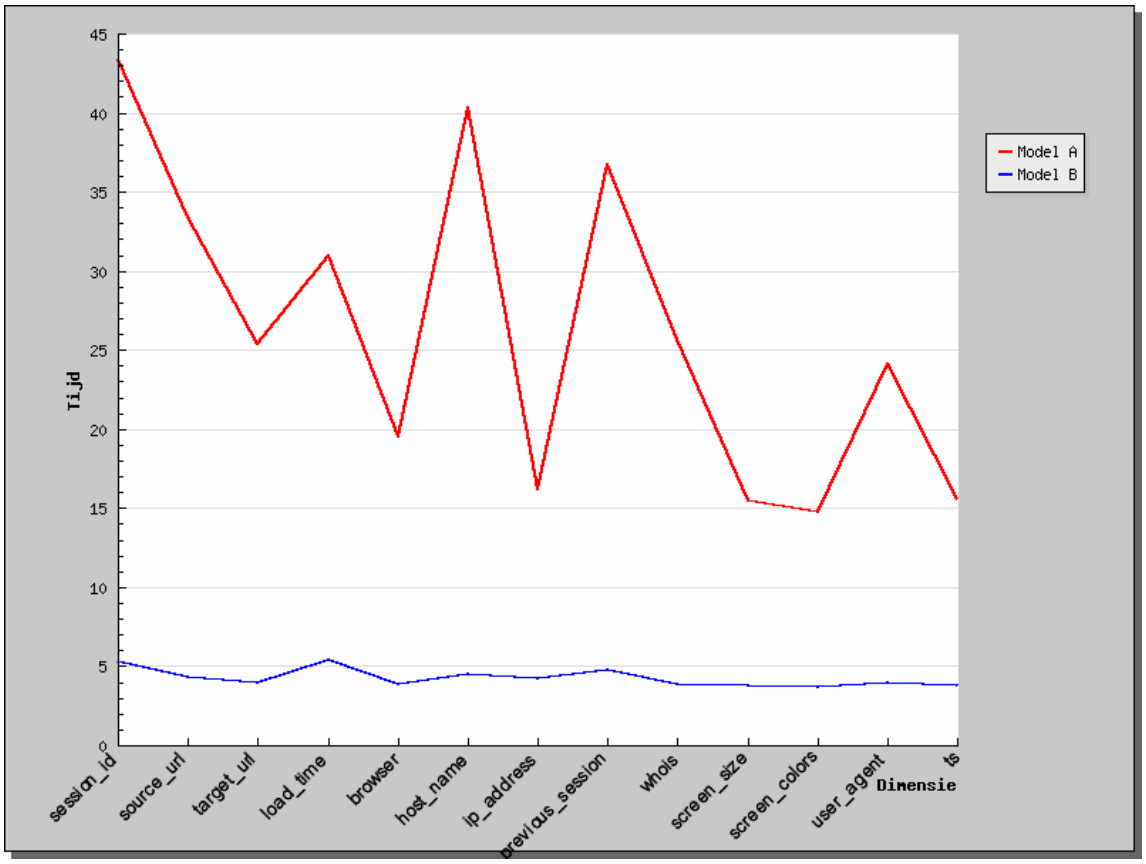
Figuur 6.5.2.5 *WOLAP Analyse van source_0250k*

source_0500k:



Figuur 6.5.2.6 *WOLAP Analyse van source_0500k*

source_1000k:



Figuur 6.5.2.7 *WOLAP Analyse van source_1000k, alleen Model B*

Als algemene tendens valt op dat hoe groter de dataset hoe extremer de optimalisatie zich laat uitbetalen in tijd. De verschillende dimensies blijven echter per model ongeveer gelijk, deze stijgen alleen mee in tijd met het aantal records. De extra tijd lijkt redelijk klein te zijn terwijl dit in model A een veel steilere curve oplevert.

Tot slot dienen we te kijken of de testresultaten de 3-deling van hoofdstuk 3 ondersteunen.

Nogmaals de 3-deling:

Methode	Voordelen	Nadelen
1. Geen transformatie (Model A)	• Geen overhead bij gewoon gebruik (*insert / update / select*) • Maar 1 tabel nodig	• Doorsnijdingen maken is erg traag • Dimensie opdeling is niet mogelijk in de database omgeving • Reëel gezien niet te gebruiken voor *OLAP* functionaliteit
2. Starscheme transformatie, zonder extra opdeling in dimensies (Model B)	• Relatief gemakkelijk incrementeel bij te houden • Doorsnijdingen maken een stuk sneller dan een platte tabel • Voor kleinere datasets te gebruiken voor *OLAP* functionaliteit	• Overhead bij gewoon gebruik • Dimensie opdeling wordt hier niet gebruikt • 1 + #dimensies (zichtbaar in de originele dataset) tabellen
3. Starscheme transformatie (Model C)	• Goed te gebruiken voor *OLAP* functionaliteit • Snelste doorsnijding (1) • Dimensie opdeling gaat zover de gebruiker wil	• Veel overhead bij gewoon gebruik (2) • Moeilijker incrementeel bij te houden • 1 + #dimensies (zichtbaar in de originele dataset) tabellen

O: aan de gestelde voorwaarden voldaan.
O: niet aan de gestelde voorwaarden voldaan.

1. Uit de resultaten blijkt dat model C niet altijd de snelste doorsnijding oplevert, maar wel altijd relatief even snel als model B. De winst van model C zit dus vooral in het verkrijgen van extra informatie en eventueel optimaliseren door de gebruiker.
2. Deze overhead is gelijk aan bij model B, aangezien de dimensie tabellen kunnen worden gebruikt van model B. Eventuele extra informatie wordt door het maken van model C automatisch afgehandeld, net zoals het bijhouden van deze extra informatie.

7. Conclusie

Terugkijkend en concluderend kan ik zeggen dat het onderzoek en de stage zeer geslaagd zijn. Al blijft het altijd jammer dat je niet alles kunt verwezenlijken wat je eigenlijk wilt.
De conclusie die gesteld zal worden in dit hoofdstuk zal gebeuren op basis van de testresultaten die beschreven zijn in hoofdstuk 6 en op de vraagstelling van hoofdstuk 2.

7.1 Op te leveren

De op te leveren zaken (zie hoofdstuk 2):

1) *Een duidelijk omschreven basisidee en ontwerp*
   Hieraan is voldaan door middel van deze scriptie

2) *Een model, de box, waaraan bovenstaande voorwaarden voldoen*
   Hieraan is voldaan door middel van hoofdstuk 5.

3) *Een implementatie van de datamanager*
   Hieraan is voldaan door middel van hoofdstuk 6 en bijlage A.

4) *Een proof of concept WOLAP applicatie, om de voorwaarden te kunnen testen en uitspraken te kunnen doen over het geheel. Hiermee wordt bedoeld dat er een verzameling queries uitgevoerd zullen worden op de modellen die gegenereerd worden.*
   Hieraan is voldaan door middel van hoofdstuk 6 en bijlage A.

Aan de overige voorwaarden is voldaan. Dit omdat deze zo gekozen zijn dat anders de op te leveren zaken en een gedegen conclusie onmogelijk waren.

7.2 De centrale probleemstelling

De centrale probleemstelling:

> "Is het mogelijk om een algoritme te ontwikkelen dat via datatransformaties en datamining -gebruikmakend van een niet triviale gebruikersprofilering- uitvoer geeft die als invoer kan dienen om standaard *OLAP* functionaliteit efficiënter en sneller aan te kunnen bieden via het *WWW*."

Uit de resultaten van hoofdstuk 6 blijkt dat het opsplitsen in deelmodellen en het juist toepassen van profileringinformatie en model kennis een goede methode is om te komen tot een *WOLAP* applicatie. Het integreren van de componenten en het wegpoetsen van de zijeffecten van diverse modellen vergt echter veel denkwerk.
Al met al kan denk ik gesteld worden dat het onderzoek geslaagd is en dat de centrale probleemstelling beantwoord kan worden met een volmondig "ja".

8. In de toekomst

Zoals in ieder onderzoek, heb je nooit tijd genoeg om alles te onderzoeken of uit te voeren wat je zelf voor ogen had. Aangezien het domein van het probleem zo groot is heb ik de opdracht moeten afbakenen. Dit onderzoek kan echter wel een degelijke basis vormen, mocht de box totaal geïmplementeerd gaan worden. In dit hoofdstuk zal ik puntsgewijs de elementen opnoemen waar nog flink wat werk gedaan kan worden.

De box:

1. De profilering interpreter werkt momenteel op zichzelf. De kracht van het concept zit in het kunnen aankoppelen van een datamining tool (bijvoorbeeld Safarii). Er zou ook onderzocht kunnen worden of een hybride vorm van de huidige interpreter in combinatie met een datamining tool een goede werkvorm zou opleveren.
2. Het algoritme zal ongetwijfeld efficiënter qua complexiteit opgezet kunnen worden. Een complexiteitsanalyse in combinatie met een herimplementatie van de datamanager zouden een incrementele opbouw kunnen versnellen en misschien wel zo, dat het model B minder vaak gebruikt hoeft te worden als tussenmodel.
3. De analyse van model A om een Starscheme instructie te genereren zal qua syntax en semantiek bekeken kunnen worden om een koppeling te kunnen maken met al bestaande *OLAP* applicaties, al zal deze koppeling waarschijnlijk niet gebruik kunnen maken van de profilering interpreter.
4. Het uitbreiden van model C en de bijbehorende bibliotheek aan expansie functies. Momenteel is dit gedeelte van de implementatie alleen als proof of concept ingebouwd. De kracht van de verschillende modellen zal hierdoor beter tot z'n recht komen.
5. Het inladen en heruitvoeren van instructiesets mogelijk maken in de datamanager. Het model zou dan handmatig geoptimaliseerd kunnen worden.
6. Het wegwerken van de nadelen van de 3-deling wat betreft het bijhouden van de datamodellen. Dit zou in het Framework van Efficiency Online automatisch bijgehouden kunnen worden. Dit vergt echter een onderzoek en een gedegen implementatie.

WOLAP:

1. Een complete implementatie van een *WOLAP* applicatie, die de box snapt.
2. Bij het uitvoeren van punt *WOLAP*:1 zal er misschien gekozen kunnen worden voor een tussen API zodat punt De box:3 gemakkelijker tot stand komt. Ook het bouwen van een *WOLAP* applicatie zal dan in complexiteit afnemen.

Literatuurlijst

De literatuurlijst die is gebruikt voor het maken van deze scriptie

1. **Graph Transformation for Specifcation and Programming**
   Marc Andries, Gregor Engels, Annegret Habel, Bertholf Hoffman, Hans-Jörg Kreowski, Sabine Kuske, Detlef Plump, Andy Schürr, Garbiele Taentzer
   Science of Computer Programming, Vol. 34, No. 1, April 1999, pp.1-54

2. **Discrete Mathematics with Graph Theory**
   Edgar G. Goodaire, Michael M. Parmente
   Prentice Hall, 1998

3. **Wikipedia**
   http://explanation-guide.info/meaning/OLAP.html
   Wikipedia onder de voorwaarden van GNU FDL

4. **Internetwoordenboek**
   http://internetwoordenboek.kennisnet.nl, Stichting Kennisnet

5. **Multi-Relational Data Mining**
   Arno Knobbe
   Kiminkii, Proefschrift, o.t.v. SIKS, 2004

6. **Graph Transformation in Relational Databases**
   Gergely Varró, Katalin Friedl, Dániel Varró
   Proc. GraBaTs 2004: International Workshop on Graph Based Tools

7. **Performance-based item analysis for profiling skills and competencies for progress files**
   Gergely Varró, Katalin Friedl, Dániel Varró
   University of Luton, 2002

8. **MySQL Reference Manual**
   http://dev.mysql.com/doc/mysql/en/storage-requirements.html, MySQL sectie Developer Zone

9. **OLAP Standaarden en ontwikkelingen**
   http://xml.coverpages.org/xmlForAnalysis.html, Cover Pages sectie Technology Reports
   http://www.databasejournal.com/features/mssql/article.php/1495511, Database Journal sectie MS SQL.

10. **Transformation-Based Database Engineering**
    Jean-Luc Hainaut, Patrick van Bommel (Editor)
    Chapter 1, Transformation of knowledge, information and data, 2004.

Bijlage A, Documentatie van de implementatie

Deze bijlage bestaat uit de broncode. Deze is voorzien van commentaar volgens de standaarden van Efficiency Online. Deze is ontdaan van alle Framework informatie en overbodige aanroep informatie.

Deze bijlage is gegenereerd met behulp van phpDocumentor RC 1.3.0.

Class: EOLDBAction

Source Location: /scriptie/classes/EOLDBAction.php

Class Overview **[line 9]**

Class with holds Database, Table & Field Calls

Author(s):
Version:
Copyright:

Variables

- $button
- $db
- $known_libs
- $tablename
- $total_rows

Constants

Methods

- EOLDBAction
- buildHybride
- buildInstruction
- buildModelB
- buildModelC
- buildModelCPre
- clearInstructions
- fillDimensionTable
- fillModelB
- fillModelBWithDim
- getFieldInformation
- getFieldUnicity
- getInstructionList
- getPreModelCList
- getRealMaxFieldSize
- getSummary
- getTotalRows
- handleSummary
- setParameters
- setTotalRows
- testModelB
- translateType

Class Details

Class with holds Database, Table & Field Calls

Class Variables

$button = *null*

[line 12]

Type: mixed

Overrides:

$db = *null*

[line 11]

Type: mixed

Overrides:

$known_libs = *array('datetime' => 'expandDatetime')*

[line 571]

Type: mixed

Overrides:

$tablename = *''*

[line 13]

Type: mixed

Overrides:

$total_rows = 0

[line 14]

Type: mixed

Overrides:

Class Methods

EOLDBAction

```
EOLDBAction EOLDBAction( string $tablename)
```

[line 21]

Constructor

Parameters:

- string **$tablename** - a tablename

---

buildHybride

```
string buildHybride( EOLWebApp &$sys)
```

[line 659]

Builds the hybride model

Tags:

- **return** - HTML

Parameters:

- EOLWebApp **&$sys** - the current system instance

---

buildInstruction

```
string buildInstruction( EOLWebApp &$sys)
```

[line 263]

Build instructionset based on the selected table (model A)

Tags:

- **return** - Instructionset build summary

Parameters:

- EOLWebApp **&$sys** - the current system instance

---

buildModelB

```
string buildModelB( EOLWebApp &$sys)
```

[line 410]

Builds model B

Tags:

- **return** - HTML

Parameters:

- EOLWebApp **&$sys** - the current system instance

---

buildModelC

```
string buildModelC( EOLWebApp &$sys)
```

[line 596]

Builds model C

Tags:

- **return** - HTML

Parameters:

- EOLWebApp **&$sys** - the current system instance

---

buildModelCPre

```
string buildModelCPre( EOLWebApp &$sys)
```

[line 578]

Build the model C analyses

Tags:

- **return** - HTML

Parameters:

- EOLWebApp **&$sys** - the current system instance

---

clearInstructions

```
void clearInstructions( )
```

[line 241]

Delete all the instructions for the currently selected table

Parameters:

---

fillDimensionTable

```
Boolean fillDimensionTable( string $dim_table, string $fieldname)
```

[line 291]

Fills a dimensiontable

Parameters:

- string **$dim_table** - a dimension table
- string **$fieldname** - a fieldname

---

fillModelB

```
string fillModelB( array &$rules, string &$target_table, [string
$model_name = "B"])
```

[line 340]

Fill model B with the plain data (no forein fields)

Tags:

- **return** - HTML

Parameters:

- array **$rules** - the instructions
- string **$target_table** - a target table
- string **$model_name** - a modelname

---

fillModelBWithDim

```
string fillModelBWithDim( array &$queries, string &$target_table,
[string $model_name = "B"])
```

[line 308]

Fills the empty dimension fields in a Model with the correct information

Tags:

- **return** - HTML

Parameters:

- array **$queries** - alle the querie templates to execute
- string **$target_table** - a target table
- string **$model_name** - a modelname

---

getFieldInformation

```
array getFieldInformation( all 0)
```

[line 100]

Get all the field information from all fields in a table

Tags:

- **return** -

of objects containing [columname] => object(field information):

$object->blob $object->max_length $object->multiple_key $object->name $object->not_null $object->numeric $object->primary_key $object->table $object->type $object->unique_key $object->unsigned $object->zerofill

Parameters:

- all **0** - global

---

getFieldUnicity

```
double getFieldUnicity( string $fieldname)
```

[line 125]

Get field unicity from a field in a table

Tags:

- **return** - percentage

Parameters:

- string **$fieldname** - a fieldname

---

getInstructionList

```
string getInstructionList( current &$rules)
```

[line 217]

display the instructonset

Tags:

- **return** - which holds the summary (HTML)

Parameters:

- current **&$rules** - created rules (instructions)

---

getPreModelCList

```
string getPreModelCList( array &$rules, array &$result)
```

[line 537]

Create the tests on model B

Tags:

- **return** - HTML

Parameters:

- array **$rules** - the instructions
- array **$result** - the result

getRealMaxFieldSize

```
int getRealMaxFieldSize( string $fieldname)
```

[line 142]

Get the real maximum fieldsize

Tags:

- **return** - real maximum fieldsize

Parameters:

- string **$fieldname** - a fieldname

getSummary

```
string getSummary( EOLWebApp &$sys)
```

[line 192]

Display the summary

Tags:

- **return** - which holds the summary (HTML)

Parameters:

- EOLWebApp **&$sys** - the current system instance

getTotalRows

```
int getTotalRows( )
```

[line 78]

Get the total rows that where calculated

Tags:

- **return** - Total nr of rows

Parameters:

---

handleSummary

```
Boolean handleSummary( )
```

[line 158]

Handle action from the displayed table summary

Parameters:

---

setParameters

```
Boolean setParameters( string $tablename)
```

[line 35]

Set parameters in a non framework way

Parameters:

- string **$tablename** - a tablename

---

setTotalRows

```
Boolean setTotalRows( )
```

[line 61]

Get total nr of rows in the current table

Parameters:

---

testModelB

```
array testModelB( array &$rules)
```

[line 471]

Tests model B

Tags:

- **return** - result

Parameters:

- array **$rules** - the instructions

---

translateType

```
string translateType( string $type)
```

[line 178]

Translate the MySQL PHP fieldtype return string to a MySQL fieldtype

Tags:

- **return** - a fieldtype

Parameters:

- string **$type** - a fieldtype

Class: EOLSystem

Source Location: /scriptie/classes/EOLSystem.php

Class Overview **[line 6]**

Class with holds Application Framework Calls wich are usefull

Author(s):
Version:
Copyright:

Variables

- $button

Constants

Methods

- EOLSystem
- getQuestion
- getSelectionDropDown
- getSelectTablename
- getSessionVar
- getSystemVar
- getTableDropdown
- getVar
- handleBuild
- handleGoto
- handleQuestionaire
- resetProgress
- setSessionVar

Class Details

Class with holds Application Framework Calls wich are usefull

---

Class Variables

$button =

[line 8]

Type: mixed

Overrides:

Class Methods

EOLSystem

```
EOLSystem EOLSystem( EOLButtonFactory $button)
```

[line 14]

Constructor

Parameters:

- EOLButtonFactory **$button** - an instance of the button creator

---

getQuestion

```
HTML getQuestion( EOLWebApp &$sys)
```

[line 145]

Get the question based on the progress

Parameters:

- EOLWebApp **&$sys** - the current system instance

---

getSelectionDropDown

```
HTMLDropDown getSelectionDropDown( int $tableconf_id, mixed
$default, string $name)
```

[line 87]

Get Dropdown

Parameters:

- int **$tableconf_id** - un_tableconfiguration_id
- mixed **$default** - default value to select
- string **$name** - name of the dropdown

---

getSelectTablename

```
HTML getSelectTablename( EOLWebApp &$sys)
```

[line 121]

Get the selected tablename or the dropdown with all the possible model A tables

Parameters:

- EOLWebApp **&$sys** - the current system instance

---

getSessionVar

```
mixed getSessionVar( string $var)
```

[line 39]

Getting a variable from the systems session environments

Tags:

- **return** - value

Parameters:

- string **$var** - a variable name

---

getSystemVar

```
mixed getSystemVar( string $var)
```

[line 67]

Generic function for getting a variable from the systems postlist mechanism

If it exists in the postlist, store it in the session Also checks the systems session for the variable

Tags:

- **return** - value

Parameters:

- string **$var** - a variable name

---

getTableDropdown

```
HTMLDropDown getTableDropdown( EOLWebApp &$sys, string $tablename,
mixed $default, string $name)
```

[line 112]

Get Dropdown from tablename

Parameters:

- EOLWebApp **&$sys** - the current system instance
- string **$tablename** - a tablename

- mixed **$default** - default value to select
- string **$name** - name of the dropdown

---

getVar

```
mixed getVar( string $var)
```

[line 25]

Get a variable from the systems post / get environment

Tags:

- **return** - value

Parameters:

- string **$var** - a variable name

---

handleBuild

```
none handleBuild( EOLWebApp &$sys, string $build)
```

[line 215]

Handle a build action

Parameters:

- EOLWebApp **&$sys** - the current system instance
- string **$build** - the action to be build

---

handleGoto

```
HTML handleGoto( EOLWebApp &$sys, string $goto)
```

[line 251]

Handle a goto action

Parameters:

- EOLWebApp **&$sys** - the current system instance
- string **$goto** - goto action

---

handleQuestionaire

```
HTML handleQuestionaire( EOLWebApp &$sys)
```

[line 190]

Handle the current answer to the previous question

Parameters:

- EOLWebApp **&$sys** - the current system instance

---

resetProgress

```
void resetProgress( EOLWebApp &$sys)
```

[line 271]

Reset all the stored session variables

Parameters:

- EOLWebApp **&$sys** - the current system instance

---

setSessionVar

```
void setSessionVar( string $var, string $val)
```

[line 53]

Store a variable in the session

Parameters:

- string **$var** - a variable name
- string **$val** - a variable value

Class: EOLTable

Source Location: /scriptie/classes/EOLTable.php

Class Overview **[line 9]**

Class with holds Table & Field create calls

Author(s):
Version:
Copyright:

Variables

- $query

Constants

Methods

- EOLTable
- addFieldCreateTable
- addIndex
- addUnique
- closeCreateTable
- createTable
- openCreateTable

Class Details

Class with holds Table & Field create calls

This class contains also some index and unique key features

Class Variables

$query = ""

[line 11]

Type: mixed

Overrides:

Class Methods

EOLTable

```
EOLTable EOLTable( )
```

[line 16]

Constructor

Parameters:

---

addFieldCreateTable

```
none addFieldCreateTable( string $fieldname, string $type, int
$length)
```

[line 40]

Add a field to a current

Parameters:

- string **$fieldname** - a fieldname
- string **$type** - a fieldtype
- int **$length** - the maximum fieldlenght

---

addIndex

```
Boolean addIndex( MySQLResource &$db, string &$tablename, string
&$fieldname)
```

[line 92]

Add a index on a field

Parameters:

- MySQLResource **&$db** - a MySQL resource link
- string **$tablename** - a tablename
- string **$fieldname** - a fieldname

---

addUnique

```
Boolean addUnique( MySQLResource &$db, string &$tablename, string
&$fieldname)
```

[line 75]

Add a unique key on a field

Parameters:

- MySQLResource **&$db** - a MySQL resource link
- string **$tablename** - a tablename
- string **$fieldname** - a fieldname

---

closeCreateTable

```
Boolean closeCreateTable( )
```

[line 60]

Close the current create table statement

Parameters:

---

createTable

```
Boolean createTable( MySQLResource &$db)
```

[line 107]

Execute the builded create table statement

Parameters:

- MySQLResource **&$db** - a MySQL resource link

---

openCreateTable

```
Boolean openCreateTable( string $tablename)
```

[line 25]

Generate start of a create table statement

Parameters:

- string **$tablename** - a tablename

Class: ScrInstruction

Source Location: /scriptie/classes/ScrInstruction.php

Class Overview **[line 9]**

Class with holds an instruction, there will be the # of columns

Author(s):
Version:
Copyright:

Variables

- $fieldname
- $fieldtype
- $label
- $lib_function
- $make_foreign
- $max_fieldlength
- $real_max_fieldlength
- $tablename
- $unicity
- $un_fieldconfiguration_id
- $un_tableconfiguration_id

Constants

Methods

- ScrInstruction
- fillMakeForeign
- saveInstruction
- setFieldConf
- setFieldname
- setLabel
- setTableConf
- setTablename

Class Details

Class with holds an instruction, there will be the # of columns

instances active during the proces

Class Variables

$fieldname = ''

[line 23]

Type: mixed

Overrides:

$fieldtype = 'f'

[line 15]

Type: mixed

Overrides:

$label = ''

[line 12]

Type: mixed

Overrides:

$lib_function = ''

[line 25]

Type: mixed

Overrides:

$make_foreign = 'N'

[line 19]

Type: mixed

Overrides:

$max_fieldlength = 0

[line 16]

Type: mixed

Overrides:

$real_max_fieldlength = 0

[line 17]

Type: mixed

Overrides:

$tablename = ''

[line 22]

Type: mixed

Overrides:

$unicity = 0

[line 18]

Type: mixed

Overrides:

$un_fieldconfiguration_id = 0

[line 14]

Type: mixed

Overrides:

$un_tableconfiguration_id = 0

[line 13]

Type: mixed

Overrides:

Class Methods

ScrInstruction

```
ScrInstruction ScrInstruction( string $tablename, string
$fieldname)
```

[line 32]

Constructor

Parameters:

- string **$tablename** - a tablename
- string **$fieldname** - a fieldname

---

fillMakeForeign

```
void fillMakeForeign( )
```

[line 101]

Set the make foreign variable based on the fieldtype en fieldlength

Parameters:

---

saveInstruction

```
void saveInstruction( )
```

[line 116]

Write the current object to the database, interface for latter use

Parameters:

---

setFieldConf

```
Boolean setFieldConf( )
```

[line 87]

Set fieldconfiguration

Parameters:

---

setFieldname

```
Boolean setFieldname( string $var)
```

[line 52]

Set fieldname

Parameters:

- string **$var** - a fieldname

---

setLabel

```
Boolean setLabel( )
```

[line 61]

Contructs label from the earlier set tablename & fieldname

Parameters:

---

setTableConf

```
Boolean setTableConf( )
```

[line 73]

Set the un_tableconfiguration id from the system

Parameters:

---

setTablename

```
Boolean setTablename( string $var)
```

[line 42]

Set tablename

Parameters:

- string **$var** - a tablename

Class: ScrLib

Source Location: /scriptie/classes/ScrLib.php

Class Overview **[line 8]**

Class which holds the library functions for expanding certain columns

Author(s):
Version:
Copyright:

Variables

Constants

Methods

- ScrLib
- expandDatetime

Class Details

Class which holds the library functions for expanding certain columns

---

Class Variables

Class Methods

ScrLib

```
ScrLib ScrLib( )
```

[line 13]

Constructor

Parameters:

---

expandDatetime

```
Boolean expandDatetime( MySQLResource &$db, string &$tablename,
string &$fieldname)
```

[line 24]

Expand a datetime field

Parameters:

- MySQLResource **&$db** - a MySQL resource link
- string **$tablename** - a tablename
- string **$fieldname** - a fieldname