SOCIAL DBMS

ONDERZOEK naar het meest geschikte

DATABASEMANAGEMENTSYSTEEM

voor de opslag van grote hoeveelheden SOCIALE DATA

AFSTUDEERDOSSIER DOOR NICK GROENEWEGEN (60190)

NCIM GROEP

DE HAAGSE HOGESCHOOL

SocialDBMS
Afstudeerdossier

Onderzoek naar het meest geschikte database-managementsysteem voor de opslag van grote hoeveelheden sociale data.

Door: Nick Groenewegen
Studentnummer: 60190
Studie: Technische Informatica op de Haagse Hogeschool Delft

Bedrijf: NCIM-Groep
Bedrijfsmentor: Dhr. M. Streppel

1e examinator: Dhr. H. van den Bosch
2e examinator: Dhr. T. Andrioli

Datum: 31-05-2011
Versie 1.0

Inhoud afstudeerdossier

Het afstudeerdossier bestaat uit het ***Afstudeerplan***, het ***Plan van aanpak*** en het ***Onderzoeks-rapport***. De scriptie is apart ingebonden, zodat de scriptie en het afstudeerdossier naast elkaar bekeken kunnen worden.

SocialDBMS
Afstudeerplan

Onderzoek naar het meest geschikte database-managementsysteem voor de opslag van grote hoeveelheden sociale data.

Door: Nick Groenewegen
Studentnummer: 60190
Studie: Technische Informatica op de Haagse Hogeschool Delft

Bedrijf: NCIM-Groep
Bedrijfsmentor: Dhr. M. Streppel

1e examinator: Dhr. H. van den Bosch
2e examinator: Dhr. T. Andrioli

Datum: 31-05-2011
Versie 2.0

Afstudeerplan

Informatie afstudeerder en gastbedrijf (*structuur niet wijzigen*)

Afstudeerblok: 2011-1.1 (start uiterlijk 7 februari 2011)
Startdatum uitvoering afstudeeropdracht: 10-01-2011
Einddatum uitvoering afstudeeropdracht: 13-05-2011
Inleverdatum afstudeerdossier volgens jaarrooster: 6 juni 2011

Studentnummer: 60190
Achternaam: dhr Groenewegen
Voorletters: N.T.P.
Roepnaam: Nick
Adres: Kreekrug 8
Postcode: 2678 PR
Woonplaats: De Lier
Telefoonnummer: 0174-521684
Mobiel nummer: 0631975538
Privé emailadres: groenewegen.nick@gmail.com

Opleiding: Technische Informatica
Locatie: Delft
Variant: voltijd

Naam studieloopbaanbegeleider: Elise Noordtzij
Naam begeleider/examinator: Henk van den Bosch
Naam expert/examinator: Toni Andrioli

Naam bedrijf: NCIM-Groep
Afdeling bedrijf: Java
Bezoekadres bedrijf: Veurse Achterweg 26
Postcode bezoekadres: 2264 SG
Postbusnummer:
Postcode postbusnummer:
Plaats: Leidschendam
Telefoon bedrijf: 070-3501554
Telefax bedrijf: 070-3545515
Internetsite bedrijf: http://www.ncim-groep.nl

Achternaam opdrachtgever: dhr. Kasperink
Voorletters opdrachtgever*:* H.
Titulatuur opdrachtgever: MBA BSc
Functie opdrachtgever: Uitvoerend Vice-President en COO
Doorkiesnummer opdrachtgever:
Email opdrachtgever: h.kasperink@ncim.nl

Achternaam bedrijfsmentor: dhr. Streppel
Voorletters bedrijfsmentor: M.
Titulatuur bedrijfsmentor: Dr.
Functie bedrijfsmentor: Senior software developer
Doorkiesnummer bedrijfsmentor: 06-46316984
Email bedrijfsmentor: m.streppel@ncim.nl

Doorkiesnummer afstudeerder*:*
Functie afstudeerder (deeltijd/duaal): voltijd

Titel afstudeeropdracht:
Onderzoek naar meest geschikte databasemanagementsysteem voor massa opslag van data.

Opdrachtomschrijving

1. Bedrijf

De NCIM Groep is gespecialiseerd in het detacheren van goed opgeleide professionals voor projectmanagement, consulting, softwareontwikkeling en system management in technische omgevingen.

De NCIM Groep bestaat sinds 1988 en heeft vestigingen in Leidschendam, Eindhoven, Brussel en Camberly (U.K) en Fort Worth (U.S.A). Het hoofdkantoor is gevestigd in Leidschendam.

Meer dan 200 bekwame professionals met een passie voor technologie bouwen namens NCIM aan ICT-oplossingen voor opdrachtgevers in de sectoren Defensie en Veiligheid, Energie en utilities, Telecommunicatie, transport en media, en Technische automatisering.

2. Probleemstelling

Mensen zetten steeds meer (privé) informatie op internet. Vooral de sociale netwerken zijn populair, zowel zakelijk als privé. Denk hierbij aan Hyves, Twitter, LinkedIn en Facebook.

Deze bronnen van informatie bieden een mogelijkheid op het gebied van forensisch onderzoek. Personen kunnen onderling aan elkaar gekoppeld worden waardoor extra opsporingsmogelijkheden ontstaan.

NCIM werkt samen met klanten uit de Defensie en Veiligheid sector. Voor deze sector is het interessant om te weten wat de mogelijkheden zijn op het gebied van verzamelen van gegevens van sociale netwerken met betrekking tot forensisch onderzoek. Momenteel kan NCIM hen alleen een uitleg geven over de mogelijkheden van deze nieuwe informatiebron. Het probleem: er mist een prototype waarmee NCIM de mogelijkheden aan (potentiële) klanten kan demonstreren.

Om dit probleem op te lossen is er een intern project gestart: het SNScanner project. Momenteel werken er drie softwareontwikkelaars van NCIM aan de bouw van een prototype voor het SNScanner project. Omdat het buiten de scope van het SNScanner project valt, is er tijdens de ontwikkeling geen onderzoek gedaan naar geschikte databasemanagementsystemen (DBMS).

Door het ontbrekende DBMS onderzoek in het SNScanner project is het voor NCIM nog onduidelijk welke DBMS-en geschikt zijn voor de opslag van grote hoeveelheden *sociale data*[1].

[1] Als we spreken over 'sociale data' wordt hiermee de data bedoelt die verzameld is van de verschillende sociale netwerken.

3. Doelstelling van de afstudeeropdracht

De aanleiding van deze opdracht is het missende DBMS onderzoek tijdens het SNScanner project. Het SNScanner project bestaat uit de volgende onderdelen:

1. Het ontwikkelen van crawlers die de gegevens van sociale netwerken verzamelen en onderling koppelen.
2. De implementatie van een DBMS die de sociale data, geleverd via de crawlers (1), opslaat en beschikbaar stelt voor presentatie (3).
3. Het ontwikkelen van een applicatie die de grote hoeveelheden opgeslagen 'sociale data' op overzichtelijke wijze kan presenteren.

Doelstelling SNScanner voor NCIM

- Het kunnen demonsteren van een prototype aan klanten in de Defensie en Veiligheid sector. Het prototype moet de mogelijkheden van forensisch onderzoek op basis van verzamelde gegevens van sociale netwerken kunnen aantonen.

Doelstelling afstudeeropdracht voor NCIM

- Een betere DBMS keuze kunnen maken voor het SNScanner project.
- Een betere DBMS keuze kunnen maken voor toekomstige projecten aan de hand van de informatie aanwezig in een onderzoeksrapport.

Hoofddoelstelling SocialDB onderzoek

- Het documenteren van de geschiktheid van verschillende DBMS-en met betrekking tot de massa opslag en beschikbaarheid van onderling gekoppelde sociale data.

De hoofddoelstelling is op te delen in de volgende **deeldoelstellingen**:

- Het inrichten van een DBMS testomgeving.
- Het ontwikkelen van een benchmarktool die de verschillende DBMS-en kan testen.
- Het ontwikkelen van een integratieplan. Het integratieplan beschrijft de werkwijze hoe het meest geschikt bevonden DBMS in het SNScanner prototype geïntegreerd kan worden.

4. Resultaat

Het onderzoeken naar een geschikt DBMS voor de opslag van grote hoeveelheden sociale data is onderdeel van een groter project. Het resultaat van het gehele project is de ontwikkeling van een prototype wat de mogelijkheden van forensisch onderzoek, op basis van sociale netwerken, aan klanten van NCIM kan demonstreren.

Deze afstudeeropdracht richt zich op de databaselaag en heeft de volgende deelresultaten:

1. Benchmarktool die de performance van verschillende DBMS-en kan testen.
2. Integratieplan die het integratieproces beschrijft hoe het meest geschikt bevonden DBMS in het SNScanner project geïntegreerd kan worden.

Het resultaat voor NCIM bestaat uit de volgende twee onderdelen:

1. Het kunnen demonstreren van een prototype die gebruikt maakt van het meest geschikt bevonden DBMS.
2. Het DBMS rapport kunnen gebruiken bij toekomstige projecten waarbij een DBMS-keuze gemaakt moet worden.

5. Uit te voeren werkzaamheden, inclusief een globale fasering, mijlpalen en bijbehorende activiteiten

- Opbouwen afstudeerdossier (15 dagen)
- Plan van aanpak schrijven. (3 dagen)
- Het achterhalen en documenteren van de requirements. (3 dagen)
- Onderzoeken en analyseren van de gegevens van sociale netwerken die de crawlers aanleveren. (1 dag)
- Literatuuronderzoek naar beschikbare DBMS-en (15 dagen).
- Ontwikkelen van een benchmarktool. (15 dagen)
- Onderzoek doen naar geschikte DBMS-en. (15 dagen)
  - Het maken van een onderzoeksplan.
  - Verschillende benchmarks uitvoeren voor een selectie van de meest geschikte DBMS-en.
    - Capaciteit
    - Responstijd
    - Aantal connecties
  - Keuze maken van meest geschikte DBMS.
- Ontwikkelen van een integratieplan. Het integratieplan beschrijft hoe het meest geschikt bevonden DBMS in het SNScanner project geïntegreerd kan worden. (3 dagen)

6. Op te leveren (tussen)producten

- Plan van aanpak
    - Achtergrond
    - Opdrachtomschrijving
    - Aanpak
    - Projectinrichting
    - Risicofactoren
    - Kosten en baten

- DBMS onderzoeksplan
    - Huidige situatie SNScanner infrastructuur
    - Eisen aan het nieuwe DBMS
    - Doelstelling
    - Aanpak van het onderzoek
    - Afbakening onderzoek
    - Randvoorwaarden

- Benchmarktool
    - Documentatie
    - Functionele en technische beschrijving
    - Broncode

- DBMS onderzoeksrapport
    - Onderzoeksplan
    - Resultaten
    - Conclusies en aanbevelingen

- Integratieplan
    - Beschrijving hoe het meest geschikt bevonden DBMS voor het SNScanner project geïntegreerd kan worden in het SNScanner prototype.

7. Te demonstreren competenties en wijze waarop

- **A3 Achterhalen van behoeften van belanghebbenden**
  Requirements worden verzameld en gedocumenteerd in een document (Programma van eisen).

- **A4 Kiezen van een ontwikkelstrategie en ontwikkelmethodiek**
  Aan het begin van het project wordt er onderzocht welke ontwikkelstrategie en ontwikkelmethodiek het best aansluiten bij de opdracht. In het plan van aanpak wordt de keuze van ontwikkelstrategie en ontwikkelmethodiek duidelijk toegelicht.

- **C10 Ontwerpen van een systeemarchitectuur**
  Er wordt een ontwerp gemaakt van de systeemarchitectuur waarin de componenten van het systeem worden gemodelleerd.

- **D16 Het bouwen van software**
  Er wordt een benchmarktool ontwikkeld in Java. Bij het programmeren wordt er gebruik gemaakt van 'Code-' en 'Naming conventions' om leesbare en uitbreidbare code te ontwikkelen.

- **D18 Testen van een infrastructuur**
  Tijdens het afstuderen wordt er onderzoek gedaan naar de geschiktheid van verschillende DBMS-en. Na onderzoek worden de meest geschikt bevonden opties aan de hand van een testplan getest.

SocialDBMS
Plan van aanpak

Onderzoek naar het meest geschikte database-managementsysteem voor de opslag van grote hoeveelheden sociale data.

Door: Nick Groenewegen
Studentnummer: 60190
Studie: Technische Informatica op de Haagse Hogeschool Delft

Bedrijf: NCIM-Groep
Bedrijfsmentor: Dhr. M. Streppel

1e examinator: Dhr. H. van den Bosch
2e examinator: Dhr. T. Andrioli

Datum: 31-05-2011
Versie 2.0

Inhoudsopgave

1 ACHTERGROND 2

1.1 Het bedrijf 2

1.2 De opdracht 2

2 OPDRACHTOMSCHRIJVING 4

2.1 Opdrachtgever en opdrachtnemer 4

2.2 Probleemstelling 5

2.3 Doelstelling opdracht 6

2.4 Opdrachtbeschrijving 7

2.5 Op te leveren producten 8

2.6 Projectgrenzen 9

2.7 Randvoorwaarden 9

3 AANPAK 10

3.1 Methoden en technieken 10

3.2 Werkzaamheden 12

3.3 Standaarden 13

3.4 Planning 14

4 PROJECTINRICHTING 16

4.1 Projectorganisatie 16

4.2 Informatievoorziening 17

4.3 Faciliteiten 17

4.4 Kwaliteitsborging 17

5 RISICOFACTOREN 18

6 KOSTEN EN BATEN 19

7 BRONNEN 20

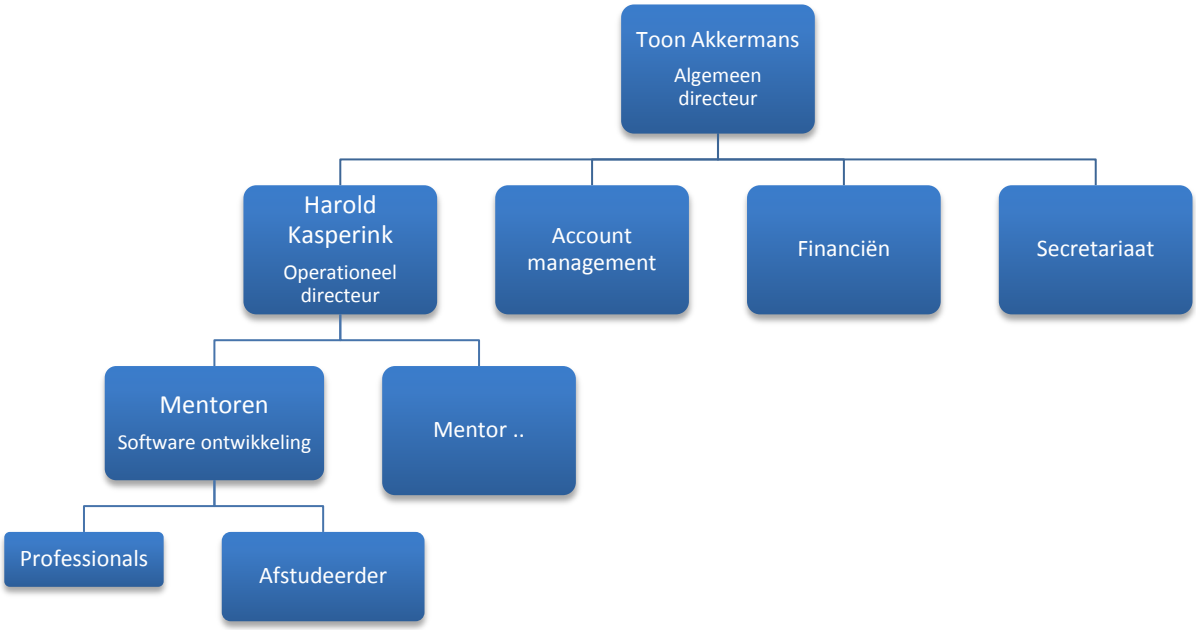
1 Achtergrond

1.1 Het bedrijf

Het project wordt uitgevoerd bij de NCIM Groep, hierna te noemen NCIM. NCIM bestaat sinds 1988 en is gespecialiseerd in het detacheren van goed opgeleide professionals voor projectmanagement, consulting, softwareontwikkeling en system management in technische omgevingen.

NCIM telt meer dan 200 professionals en heeft vestigingen in Leidschendam, Eindhoven, Brussel, Camberly (U.K) en Fort Worth (U.S.A). NCIM heeft opdrachtgevers in de sectoren Defensie en Veiligheid, Energie en Utilities, Telecommunicatie, Transport en Media, en Technische automatisering.

Dit project wordt uitgevoerd op de afdeling *softwareontwikkeling* op het hoofdkantoor in Leidschendam. Deze afdeling heeft zich gespecialiseerd in het ontwikkelen van software op basis van Microsoft, JAVA en Oracle. Een globaal organigram van de organisatie in Leidschendam is terug te vinden in figuur 1.1.1.



Figuur 1.1.1 – Organigram NCIM Groep Leidschendam

1.2 De opdracht

Dit project heeft betrekking op de sector Defensie en Veiligheid. NCIM wil klanten uit deze sector kunnen demonstreren wat de mogelijkheden zijn van forensisch onderzoek op basis van sociale netwerken.

Op dit moment werkt er een scrumteam[1] van drie personen in Leidschendam aan het project *SNScanner*. Het doel van dit project is de ontwikkeling van een webapplicatie die in staat is om gegevens te kunnen verzamelen van sociale netwerken als Hyves, Twitter, LinkedIn, YouTube en

[1] Scrum is een raamwerk voor agile management van softwareontwikkeling.

Facebook, deze gegevens op te slaan in een database en beschikbaar te maken ter presentatie in een browser.

Tijdens de ontwikkeling van het project SNScanner is er, omdat dit buiten de scope van het SNScanner project viel, geen onderzoek gedaan naar de beste databasemanagementsysteem (DBMS) keuze. Er is gekozen voor het relationele Apache Derby DBMS.

De opdracht, zoals beschreven in dit *Plan van aanpak,* komt voort uit het SNScanner project en richt zich op de database laag.

Aangezien de opdracht wordt uitgevoerd als afstudeerproject is er voor gekozen om de opdracht als onafhankelijk project uit te voeren. Ter verduidelijking: wanneer we het in dit document hebben over 'het SNScanner project', bedoelen we hier het grote overkoepelende project mee. Wanneer we het hebben over 'de opdracht', 'het project' of 'het SocialDBMS project' bedoelen we hier de afstudeeropdracht mee.

2 Opdrachtomschrijving

2.1 Opdrachtgever en opdrachtnemer

Dit project wordt uitgevoerd in opdracht van de NCIM-Groep Leidschendam en betreft een intern project. De afstudeeropdracht wordt uitgevoerd door Nick Groenewegen, Technische Informatica student bij HHS Delft.

Opdrachtgever

NCIM-Groep Leidschendam

Veurse Achterweg 26
2264 SG Leidschendam

T: +31 (0)70 - 3501554
F: +31 (0)70 - 3545515

Contactpersoon opdrachtgever / bedrijfsmentor

Micha Streppel
E: m.streppel@ncim.nl
T: +31 (0)6-46316984

Opdrachtnemer

Nick Groenewegen
Kreekrug 8
2678 PR De Lier

Studentnummer: 60190

E: groenewegen.nick@gmail.com
E: n.groenewegen@ncim.nl
T: +31 (0)6-81289240

2.2 Probleemstelling

Mensen zetten steeds meer (privé) informatie op internet. Vooral de sociale netwerken zijn populair, zowel zakelijk als privé. Denk hierbij aan Hyves, Twitter, LinkedIn en Facebook.

Deze bronnen van informatie bieden een mogelijkheid op het gebied van forensisch onderzoek. Personen kunnen onderling aan elkaar gekoppeld worden waardoor extra opsporingsmogelijkheden ontstaan.

NCIM werkt samen met klanten uit de Defensie en Veiligheid sector. Voor deze sector is het interessant om te weten wat de mogelijkheden zijn op het gebied van verzamelen van gegevens van sociale netwerken met betrekking tot forensisch onderzoek. Momenteel kan NCIM hen alleen een uitleg geven over de mogelijkheden van deze nieuwe informatiebron. Het probleem: er mist een prototype waarmee NCIM de mogelijkheden aan (potentiële) klanten kan demonstreren.

Om dit probleem op te lossen is er een intern project gestart: het SNScanner project. Momenteel werken er drie softwareontwikkelaars van NCIM aan de bouw van een prototype voor het SNScanner project. Omdat het buiten de scope van het SNScanner project valt, is er tijdens de ontwikkeling geen onderzoek gedaan naar geschikte DBMS-en.

Door het ontbrekende DBMS onderzoek in het SNScanner project is het voor NCIM nog onduidelijk wat voor DBMS-en geschikt zijn voor de opslag van grote hoeveelheden *sociale data*[2].

De opdracht, zoals beschreven in dit document, richt zich tot dit probleem. De onderzoeksvraag voor het uit te voeren onderzoek luidt:

- Wat is het meest geschikte DBMS voor de opslag van grote hoeveelheden, onderling gekoppelde, sociale data?

Deelvragen van het onderzoek worden beschreven in het onderzoeksplan. Meer informatie over de inhoud van het onderzoeksplan is te vinden in hoofdstuk 2.5.

[2] Als we spreken over 'sociale data' wordt hiermee de data bedoelt die verzameld is van de verschillende sociale netwerken.

2.3 Doelstelling opdracht

De aanleiding van deze opdracht is het missende DBMS onderzoek tijdens het SNScanner project. Het SNScanner project bestaat uit de volgende onderdelen:

1. Het ontwikkelen van crawlers die de gegevens van sociale netwerken verzamelen en onderling koppelen.
2. De implementatie van een DBMS die de sociale data, geleverd via de crawlers (1), opslaat en beschikbaar stelt voor presentatie (3).
3. Het ontwikkelen van een applicatie die de grote hoeveelheden opgeslagen 'sociale data' op overzichtelijke wijze kan presenteren.

Doelstelling opdrachtgever (SNScanner)

- Het kunnen demonsteren van een prototype aan klanten in de Defensie en Veiligheid sector. Het prototype moet de mogelijkheden van forensisch onderzoek op basis van verzamelde gegevens van sociale netwerken kunnen aantonen.

Doelstelling opdrachtgever (SocialDBMS)

- Een betere DBMS keuze kunnen maken voor het SNScanner project.
- Een betere DBMS keuze kunnen maken voor toekomstige projecten aan de hand van de informatie aanwezig in het onderzoeksrapport.

Hoofddoelstelling SocialDBMS onderzoek

- Het documenteren van de geschiktheid van verschillende DBMS-en met betrekking tot de massa opslag en beschikbaarheid van onderling gekoppelde sociale data.

De hoofddoelstelling is op te delen in de volgende **deeldoelstellingen**:

- Het inrichten van een DBMS testomgeving.
- Het ontwikkelen van een benchmarktool die de verschillende DBMS-en kan testen.
- Het ontwikkelen van een integratieplan. Het integratieplan beschrijft de werkwijze hoe de meest geschikt bevonden DBMS in het SNScanner prototype geïntegreerd kan worden.

2.4 Opdrachtbeschrijving

De uit te voeren opdracht komt voort uit het SNScanner project. Het resultaat van het SNScanner project is een prototype wat de mogelijkheden van forensisch onderzoek, op basis van sociale netwerken, aan klanten van NCIM kan demonstreren.

De SocialDBMS opdracht bestaat grotendeels uit het uitvoeren van een onderzoek naar een geschikt DBMS voor applicaties zoals het SNScanner prototype. Tijdens de uitvoering van het onderzoek wordt sociale data gebruikt zoals deze verzameld is door de crawlers van het SNScanner project. Het uit te voeren onderzoek is derhalve gericht op het onderzoeken naar geschikte DBMS-en voor het SNScanner project en toekomstige projecten waarbij met gelijksoortige data gewerkt gaat worden.

In het onderzoeksrapport worden de belangrijkste eigenschappen van de DBMS-en beschreven. Bij elk DBMS wordt toegelicht in welke situatie het DBMS het best tot ze recht komt. Door de eigenschappen en toepassingsgebieden van de DBMS-en te beschrijven is het onderzoeksrapport geschikt ter naslag voor toekomstige projecten.

Na uitvoering van het DBMS onderzoek wordt er een integratieplan ontwikkeld. Het integratieplan beschrijft de aanpak hoe het meest geschikt bevonden DBMS in het SNScanner prototype geïntegreerd kan worden.

De hoofdopdracht:

- Doe onderzoek naar geschikte DBMS-en voor de opslag en het beschikbaar maken van data vergelijkbaar met de data van het SNScanner prototype.

Het DBMS onderzoek bestaat deels uit literatuuronderzoek en deels uit het uitvoeren van een eigen experiment. Het experiment bestaat uit het benchmarken, door middel van een eigen ontwikkelde tool, van de DBMS-en.

De concrete invulling van het DBMS onderzoek wordt beschreven in het DBMS onderzoeksplan.

2.5 Op te leveren producten

Dit hoofdstuk bevat een overzicht van de op te leveren producten met een beschrijving van de inhoud van elk product.

- Plan van aanpak
    - Achtergrond
    - Opdrachtomschrijving
    - Aanpak
    - Projectinrichting
    - Risicofactoren
    - Kosten en baten

- DBMS onderzoeksplan
    - Huidige situatie SNScanner DBMS infrastructuur
    - Eisen aan het nieuwe DBMS
    - Doelstelling
    - Aanpak van het onderzoek
    - Afbakening onderzoek
    - Randvoorwaarden

- Beschrijving DBMS benchmarktool
    - Documentatie
    - Functionele en technische beschrijving

- DBMS benchmarktool
    - Broncode

- DBMS onderzoeksrapport
    - Onderzoeksplan
    - Resultaten
    - Conclusies en aanbevelingen

- Integratieplan
    - Beschrijving hoe de meest geschikt bevonden DBMS voor het SNScanner project, geïntegreerd kan worden in het SNScanner prototype.

2.6 Projectgrenzen

In dit hoofdstuk wordt de scope van de opdracht afgebakend. Hieronder een opsomming van projectgrenzen:

1. De integratie van het meest geschikt bevonden DBMS in het SNScanner prototype valt niet binnen de grenzen van dit project.
2. Tijdens het onderzoek wordt gebruik gemaakt van data aangeleverd door de crawlers van het SNScanner project. Er wordt uitgegaan van een correcte werking van de crawlers. Verbeteren van de crawlers valt buiten de grenzen van dit project.

2.7 Randvoorwaarden

Voor het succesvol uitvoeren van de opdracht dienen er tijdens de gehele afstudeerperiode aan de volgende voorwaarden te worden voldaan:

- Het project heeft een duur van 17 weken van 40 uur per week. De opdrachtgever geeft de student de benodigde tijd om binnen deze 40 uur per week te werken aan benodigde schoolvereisten.

- Tijdens de gehele afstudeerperiode staat er een docentbegeleider ter beschikking van de afstudeerder.

- Tijdens de gehele afstudeerperiode staat er een bedrijfsmentor ter beschikking van de afstudeerder. Bij de bedrijfsmentor kan de afstudeerder terecht voor vragen gerelateerd aan het stage proces, vragen van technische aard en overige vragen waarvan redelijkerwijs van mag worden aangenomen dat deze onder de verantwoordelijkheid van de bedrijfsmentor vallen.

- Het proces, de voortgang en de planning van het project wordt eenmaal per week met de bedrijfsmentor doorgenomen.

- Voor de uitvoering van het DBMS onderzoek worden de benodigde middelen ter beschikking gesteld. In het onderzoeksplan staan de benodigde middelen voor het onderzoek beschreven.

3 Aanpak

3.1 Methoden en technieken

Bij de keuze voor een projectmethodiek moet er rekening worden gehouden met de volgende projecteigenschappen / eisen:

1. Het project wordt uitgevoerd door één persoon.
2. Project duurt zeventien weken van 40 uur per week.
3. Er is een projectbegeleider die de voortgang van het project bewaakt.
4. Duur van het onderzoeken van een enkel DBMS is onbekend.

Dit project bestaat grotendeels uit onderzoek en de ontwikkeling van een benchmarktool. Tijdens het benchmarken van de verschillende DBMS-en wordt de benchmarktool steeds uitgebreid en geschikt gemaakt voor het te testen DBMS. Tijdens de onderzoeksfase kan er dus gesproken worden over iteratieve ontwikkeling van de benchmarktool en het onderzoek: na elke iteratie wordt het onderzoek completer.

Bij het maken van de keuze voor een projectmethodiek en fasering is er gebruik gemaakt van informatie beschreven in het rapport *Project Management Methodologies* (Wideman, Project Management Methodologies, 2005). In het rapport geeft Wideman een overzicht van eigenschappen van een aantal veel gebruikte methodieken. Aangezien dit project grotendeels bestaat uit het uitvoeren van onderzoek zijn de methodieken die puur bedoelt zijn voor softwareontwikkeling niet in het overzicht opgenomen. Figuur 3.1.1 geeft een overzicht van de methodiekeigenschappen gebaseerd op de eigenschappen van dit project.

Methodiek	Geschiktheid			
	1 persoon	Kort project	Iteratief	Documentatie aanwezig
PRINCE2	Nee	Nee	?	Veel
Solutions-based Project Methodology	?	Ja	?	Weinig
TenStep	Nee	Ja	Ja	Matig

Figuur 3.1.1 – Project management methodieken

Zoals figuur 3.1.1 ons vertelt sluit geen enkele methodiek compleet aan bij onze eisen. PRINCE2 is bedoelt voor projecten van grote omvang en samenwerking tussen teamleden en valt dus af. Over *Solutions-based Project Methodology* is weinig documentatie op internet beschikbaar en is dus ongeschikt voor dit project. De methodiek *TenStep* komt het dichts bij de eisen van het project. De methodiek is echter ongeschikt voor kleine projectteams (één persoon) en voldoet dus niet aan de eisen.

Aangezien er geen methodiek voorhanden is die aan de projecteisen voldoet maken we tijdens dit project gebruik van een eigen opgestelde aanpak en fasering.

Het project is aan de hand van de op te leveren producten op te delen in drie hoofdfases (figuur 3.1.2).

Figuur 3.1.2 – Globale projectfasering

De drie hoofdfases zijn onder te verdelen in de volgende globale werkzaamheden:

Figuur 3.1.3 – Concrete projectfasering

Gedetailleerde invulling van de fases is terug te vinden in hoofdstuk 3.2.

Technieken

In het SNScanner project wordt gemodelleerd met behulp van **UML**.

> "De Unified Modeling Language, afgekort UML, is een modelmatige taal om objectgeoriënteerde analyses en ontwerpen voor een informatiesysteem te kunnen maken."

Aangezien modellen van het SNScanner gebruikt gaan worden in dit project en vice versa is er voor gekozen om ook in dit project te modelleren door middel van UML.

Meer informatie over UML: http://www.uml.org/

Tijdens de requirements analyse worden eisen vastgelegd met behulp van de MoSCoW methodiek. De **MoSCoW-methode** is geschikt voor het prioriteren (Must, Should, Could, Won't have) van requirements en heeft als voordeel dat de prioriteit zelf een uitleg is van de hoogte van de prioriteit. Dit in tegenstelling tot het prioriteren op basis van cijfers.

Meer informatie over de MoSCoW-methode: http://en.wikipedia.org/wiki/MoSCoW_Method

Om tijdens het project het overzicht te kunnen behouden van de op te leveren producten, de uit te voeren werkzaamheden wordt gebruik gemaakt van *Teamwork Project Manager* (TeamworkPM). De gratis dienst van TeamworkPM biedt een webapplicatie die gebruikt kan worden voor project management. TeamworkPM biedt de volgende functionaliteiten: task management, indelen van mijlpalen, document management en time tracking.

Meer informatie over Teamwork Project Manager: http://www.teamworkpm.net/

3.2 Werkzaamheden

In dit hoofdstuk worden de verschillende fases, zoals te vinden in hoofdstuk 3, toegelicht aan de hand van de uit te voeren werkzaamheden en op te leveren producten. De planning van deze werkzaamheden is terug te vinden in hoofdstuk 3.4.

3.2.1 Analysefase

1. Plan van aanpak ontwikkelen
2. Ontwikkelen onderzoeksplan

Tijdens de analysefase staat het analyseren en beschrijven van de opdracht centraal. Aangezien de opdracht nog geen definitieve vorm kent, is het van belang om tijdens de analysefase tot een definitieve invulling te komen. Verder wordt tijdens deze fase het DBMS onderzoeksplan ontwikkeld en besproken met de opdrachtgever. Bespreking van het concept DBMS onderzoeksplan met de opdrachtgever moet er voor zorgen dat voor alle partijen duidelijk is wat het onderzoek inhoud.

- In het kort: Analyseren, requirements analyse, technisch ontwerpen.
- Op te leveren producten: Plan van aanpak, DBMS onderzoeksplan

3.2.2 Onderzoeksfase (iteratief)

1. Literatuuronderzoek naar geschikte DBMS-en.
2. Ontwikkelen benchmarktool.
3. Ontwikkelen documentatie benchmarktool
4. Testen van DBMS-en die tijdens literatuuronderzoek als geschikt naar voren komen.
5. Ontwikkelen DBMS onderzoeksrapport

Omdat er tijdens de onderzoeksfase herhalende werkzaamheden zijn, spreken we van een iteratieve aanpak. Om de iteraties gestructureerd uit te voeren wordt na elke iteratie een iteratierapport opgeleverd. Het iteratierapport bevat de evaluatie van de iteratie en beschrijft de algehele voortgang van de onderzoeksfase.

Tijdens de onderzoeksfase wordt de benchmarktool ontworpen, ontwikkeld en beschreven. Verder wordt het DBMS onderzoek uitgevoerd en het onderzoeksrapport opgesteld. Doel van deze fase is om een overzicht te krijgen van de eigenschappen van de verschillende DBMS-en.

Meer informatie over de invulling van de iteraties van de onderzoeksfase is terug te vinden in hoofdstuk 3.4.

- In het kort: Technisch ontwerpen, implementatie, testing, iteratie management.
- Op te leveren producten: Beschrijving DBMS benchmarktool, DBMS benchmarktool, DBMS onderzoeksrapport.

3.2.3 Afrondingsfase

1. Ontwikkelen DBMS onderzoeksrapport
2. Ontwikkelen integratieplan
3. Projectafsluiting

Tijdens de afrondingsfase wordt het DBMS onderzoek afgerond: het DBMS onderzoeksrapport wordt uitgebreid met de conclusies en aanbevelingen. Verder wordt tijdens de afrondingsfase het integratieplan ontwikkeld. Tot slot wordt aan het eind van de project alle producten overgedragen aan de opdrachtgever.

- Toe te passen RUP disciplines: *Deployment*
- Op te leveren producten: DBMS onderzoeksrapport, Integratieplan

3.2.4 Overige zaken

Aangezien het in dit project gaat om een afstudeeropdracht zijn er ook werkzaamheden die voor school geleverd moeten worden. Gezien deze zaken inhoudelijk niet van belang zijn voor de opdrachtgever worden deze niet verder toegelicht. Met deze zaken moet wel rekening gehouden worden met betrekking tot de planning.

De volgende werkzaamheden moeten voor school geleverd worden:

- Opbouwen afstudeerdossier
- Schrijven voortgangsverslag

3.3 Standaarden

- De bestandsnaam van op te leveren documenten is als volgt: YYYYMMDDDD OMSCHRIJVING R{X1}S{X2} – AUTEUR. {X1} wordt verhoogd in geval van inhoudelijke wijzigingen na verspreiding en {X2} in geval van tekstuele aanpassingen. Een Plan van aanpak, geschreven op 1 februari 2005, heeft de volgende bestandsnaam: 20050201 PVA R0S1.
- Documenten worden opgeleverd in pdf formaat.

3.4 Planning

De doorlooptijd van dit project bedraagt zeventien werkweken van elk 40 uur. Alle werkzaamheden, met uitzondering als de uitvoerder expliciet is benoemd, worden uitgevoerd door de opdrachtnemer zoals in hoofdstuk 2.1 beschreven.

Alle concepten, zoals benoemd in de tabellen onder 'Resultaten', worden geëvalueerd met de bedrijfsmentor en hebben als doel om de voortgang en de correctheid van de resultaten van het project te bewaken.

3.4.1 Analysefase

De analysefase wordt lineair uitgevoerd en bedraagt vijf werkweken.

Week	Werkzaamheden	Resultaten	Begindatum	Einddatum
1	- Opdrachtomschrijving opstellen - Plan van aanpak schrijven		10-01-2011	14-01-2011
2	- Plan van aanpak schrijven	- Concept plan van aanpak	17-01-2011	21-01-2011
3	- Plan van aanpak schrijven - Ontwikkelen onderzoeksplan	- Plan van aanpak	24-01-2011	28-01-2011
4	- Interviewen SNScanner projectteam - Analyseren SNScanner architectuur - Ontwikkelen onderzoeksplan	- Concept onderzoeksplan	31-01-2011	04-02-2011
5	- Ontwikkelen onderzoeksplan	- Onderzoeksplan	07-02-2011	11-02-2011

3.4.2 Onderzoeksfase

De negen weken durende onderzoeksfase wordt iteratief uitgevoerd. Voor dit project is er gekozen voor korte iteraties van één week. De onderzoeksfase is op te delen in drie onderdelen met elk een duur van drie iteraties: het literatuuronderzoek, de ontwikkeling van de benchmarktool en tot slot het testen van de DBMS-en.

Er is gekozen om met korte iteraties te werken zodat de klant elke week resultaat ziet en waar nodig kan bijsturen. Door met conceptversies te werken en elke week een iteratierapport (weekrapportage) op te leveren is de klant te allen tijde op de hoogte van de laatste projectontwikkelingen.

Week	Werkzaamheden	Resultaten	Begindatum	Einddatum
6, 7, 8	Literatuuronderzoek	- Concept onderzoeksrapport	14-02-2011	04-03-2011
9, 10, 11	Ontwikkelen benchmarktool	- Benchmarktool - Documentatie benchmarktool	07-03-2011	25-03-2011
12, 13, 14	Testen van DBMS-en	- DBMS testresultaten - Concept onderzoeksrapport	05-04-2011	22-04-2011

Een gedetailleerde planning en invulling van de werkzaamheden wordt in het onderzoeksplan vastgelegd.

3.4.2 Afrondingsfase

De afrondingsfase wordt lineair uitgevoerd, bedraagt twee werkweken plus een uitloopweek. De uitloopweek is bedoelt om eventuele vertragingen tijdens de werkzaamheden op te vangen.

Week	Werkzaamheden	Resultaten	Begindatum	Einddatum
15	- Ontwikkelen onderzoeksrapport - Ontwikkelen integratieplan	- Onderzoeksrapport	25-04-2011	29-04-2011
16	- Ontwikkelen integratieplan	- Integratieplan	02-05-2011	06-05-2011
17	- Uitloopweek		09-05-2011	13-05-2011

3.4.2 Planning school

Hieronder een overzicht van relevante schooldata.

Week of datum	Gebeurtenis
3 of 4	- Bedrijfsbezoek door de begeleider/examinator
7 of 8	- Inleveren voortgangsverslag
9 of 10	- Inleveren concept afstudeerdossier
06-06-2011 voor 12:45	- Inleveren afstudeerdossier

4 Projectinrichting

4.1 Projectorganisatie

Projectuitvoerder

Het project wordt uitgevoerd door Nick Groenewegen. Nick is verantwoordelijk voor de aanpak, planning en uitvoering van dit project.

Voor informatie over de opdrachtnemer, zie hoofdstuk 2.1.

Stage coördinator

Gezien het een afstudeerproject betreft, kunnen er vragen zijn met betrekking tot het afstuderen. Voor deze vragen kan de opdrachtnemer terecht bij Debby Voordenhout, Recruiter / Stagebegeleider bij NCIM Leidschendam.

Bedrijfsbegeleider

Micha Streppel, professional bij NCIM, is verantwoordelijk voor de inhoudelijke- en technische begeleiding van het project. Ook de voortgangsbewaking van het project behoort tot de verantwoordelijkheden van Micha Streppel.

Docentbegeleider

Vanuit de Haagse Hogeschool is er een afstudeerbegeleider toegewezen: Henk van den Bosch. Henk van den Bosch is gespecialiseerd in het ontwerpen van databases en kan dus eventueel technische vragen omtrent dit onderwerp beantwoorden. Verder is de heer van den Bosch verantwoordelijk voor de voortgang van de stage en de beoordeling van het afstuderen.

Projectleden SNScanner

Zoals eerder vermeld heeft de uit te voeren opdracht betrekking op een deel van het SNScanner project. Tijdens het verzamelen van de eisen aan het DBMS wordt de input van deze projectleden gebruikt.

Het SNScanner team bestaat uit Gertjan Al, Swen Roelofsen en David Coppens. Het team is werkzaam in het hoofdkantoor in Leidschendam.

Inhoudelijke begeleiding

Vragen betreffende de opdracht kunnen aan Harold Kasperink (Operationeel directeur) worden gesteld.

4.2 Informatievoorziening

Scrum meetings

Het SNScanner project wordt aangepakt volgens de Scrum methodiek. Elke maandag hebben de projectleden een meeting met Harold Kasperink. In deze meeting worden de voortgang en de planning besproken. Gezien de banden tussen het SNScanner project en de afstudeeropdracht is er besloten dat de opdrachtnemer van de afstudeeropdracht bij deze meetings aanwezig is. De opdrachtnemer blijft hierdoor op de hoogte van de vorderingen van het SNScanner project en krijgt de gelegenheid om tijdens de meetings database gerelateerde vragen te stellen aan de deelnemers.

Voortgangsgesprekken

Elke week heeft de opdrachtnemer een voortgangsgesprek met de bedrijfsbegeleider. In dit gesprek wordt de voortgang van de afgelopen week besproken en wordt de planning van de aankomende week bepaald.

Overige vragen

Wanneer de opdrachtnemer van de afstudeeropdracht vragen heeft over het SNScanner project, bijvoorbeeld over het DBMS, kunnen deze worden beantwoord door de SNScanner projectleden.

4.3 Faciliteiten

De volgende faciliteiten zijn benodigd tijdens het project:

- Werkplek op het hoofdkantoor in Leidschendam.
- Geschikt werksysteem voor ontwikkeling van de benchmarktool.
    - Software om Java applicaties te kunnen ontwikkelen.
- Geschikt werksysteem voor het testen van de DBMS-en.

4.4 Kwaliteitsborging

Om de kwaliteit te kunnen waarborgen worden de volgende maatregelen genomen:

- Op te leveren documenten worden gecontroleerd door de bedrijfsbegeleider en Harold Kasperink.
    - Documenten kunnen aan de hand van geleverde feedback verbeterd worden.
    - Door deze controle uit te voeren, is het voor alle partijen duidelijk wat de voortgang en planning van het project is. Wanneer benodigd kan het project tijdig bijgestuurd worden.
- Om kwaliteitsbroncode te kunnen ontwikkelen wordt er gebruikt gemaakt van *Coding standards* en *Naming Conventions* zoals beschreven in de whitepaper: *Java™ Coding Style Guide* door Achut Reddy, Sun Microsystems, Inc. 1998.

5 Risicofactoren

Bij projecten die langere tijd lopen, is er altijd een gevaar dat de voortgang van het project risico loopt. Dit kunnen beheersbare maar soms ook onbeheersbare risico's zijn. In dit hoofdstuk zijn de aan dit project verbonden risico's beschreven.

Ontwikkeling SNScanner prototype niet voltooid

Dit project is deels afhankelijk van het SNScanner project. Het SNScanner project is een interne opdracht waaraan medewerkers van NCIM werken die tijdelijk niet gedetacheerd zijn. Voor deze medewerkers kunnen echter op elk moment opdrachtgevers gevonden worden. Wanneer projectleden van het SNScanner project tussentijds gedetacheerd worden, kan de voortgang van het SNScanner project in gevaar komen.

- **Gevolg:** Het integratieplan, wat de werkwijze van de DBMS integratie beschrijft in het SNScanner project, is niet gebaseerd op het definitieve prototype.
- **Kans:** Groot
- **Impact:** Laag
- **Tegenmaatregel:** Integratieplan word beschreven op basis van het niet definitieve prototype.

Vertraging tijdens project

Elk project kan vertraging oplopen. Dit kan bijvoorbeeld zijn door ziekte, onvoorziene werkzaamheden of andere tegenvallers.

- **Gevolg:** De voortgang van het project loopt gevaar. Project kan niet tijdig worden afgerond.
- **Kans:** Groot
- **Impact:** Midden
- **Tegenmaatregel:** Aan het eind van het project is een uitloopweek ingepland. Ook kan er tijdens één van de iteraties besloten worden om bijvoorbeeld een minder belangrijke test te laten vallen. Dit alles in overleg met de opdrachtgever.

Eisen aan het SNScanner DBMS veranderen

De conclusies en aanbevelingen van het onderzoek zijn deels afhankelijk van het SNScanner project. Tijdens het SNScanner project kunnen de eisen aan het DBMS veranderen. Wanneer de eisen aan het SNScanner DBMS veranderen kan het zijn dat de conclusies en aanbevelingen niet overeenkomen met de veranderde situatie.

- **Gevolg:** Conclusies en aanbevelingen in het onderzoeksrapport komen niet overeen met de veranderde situatie.
- **Kans:** Midden
- **Impact:** Midden
- **Tegenmaatregel:** In het onderzoeksrapport worden conclusies en aanbevelingen gedaan aan de hand van verschillende situaties. Door verschillende situaties te beschrijven is het opgestelde onderzoeksrapport relevant voor het SNScanner prototype wanneer eisen veranderen en voor andere licht afwijkende toekomstige projecten.

6 Kosten en baten

Dit hoofdstuk geeft een opsomming van de kosten en baten met betrekking tot dit project. Bij de kosten en baten zijn geen bedragen gegeven als deze intern bij de opdrachtgever bekend zijn of wanneer deze niet direct in geld zijn uit te drukken.

6.1 Kosten

- Inrichten en onderhouden van werkplek opdrachtnemer
- Inwerken van opdrachtnemer
- Beschikbaar stellen van een begeleider (NCIM professional)
- Salaris opdrachtnemer

6.2 Baten

- Er is kennis over de DBMS-en gedocumenteerd in het onderzoeksrapport. De informatie in het onderzoeksrapport kan gebruikt worden voor toekomstige projecten. Bij toekomstige projecten kan de aanwezige informatie tijd besparen.
- Het SNScanner prototype wordt kwalitatief beter en geeft NCIM een beter demonstratieprototype. Klanten worden gebonden door kwalitatieve producten en zijn dus eerder geïnteresseerd in samenwerking met NCIM.

7 Bronnen

7.1 Bronnen

Unified Modeling Language - Wikipedia. (2010, november 25). Opgeroepen op januari 25, 2011, van Wikipedia: http://nl.wikipedia.org/wiki/Unified_Modeling_Language

Wideman, R. M. (2005). *Project Management Methodologies.* Opgeroepen op 01 20, 2011, van Max's: http://www.maxwideman.com/papers/pm-methodologies/pm_methodologies.pdf

SocialDBMS
Onderzoeksrapport

Onderzoek naar het meest geschikte database-managementsysteem voor de opslag van grote hoeveelheden sociale data.

Door: Nick Groenewegen
Studentnummer: 60190
Studie: Technische Informatica op de Haagse Hogeschool Delft

Bedrijf: NCIM-Groep
Bedrijfsmentor: Dhr. M. Streppel

1e examinator: Dhr. H. van den Bosch
2e examinator: Dhr. T. Andrioli

Datum: 31-05-2011
Versie 2.0

Inhoudsopgave

1 INTRODUCTIE 3

1.1 Inleiding 3
1.2 Probleemstelling 4
1.3 Doelstelling 4
1.4 Startpunt 4

2 ONDERZOEKSPLAN 5

2.1 Probleemanalyse: het Social Network Scanner project 5
2.2 Totstandkoming van het onderzoek 6
2.3 Planning 8
2.4 Invulling literatuuronderzoek 9
2.5 Invulling benchmarktool ontwikkeling 12
2.6 Invulling performancetest 15

3 VERSCHILLENDE DATAMODELLEN 19

3.1 Het hiërarchisch model 19
3.2 Het netwerk model 19
3.3 Het relationele model 20
3.4 Object-oriented model 21
3.5 NoSQL 21

4 HET DBMS GEBRUIK BIJ DE SOCIALE NETWERKEN 24

4.1 Doelstelling en Aanpak 24
4.2 Resultaten 25
4.3 Evaluatie 27

5 OVERZICHT DATABASEMANAGEMENTSYSTEMEN 28

5.1 Derby 28
5.2 MySQL 30
5.3 HBase 32
5.4 Apache Cassandra 34
5.5 Project Voldemort 36
5.6 Oracle Database 38
5.7 Neo4J 40
5.8 MongoDB 42
5.9 PostgreSQL 44
5.10 Microsoft SQL Server 46
5.11 Apache CouchDB 48
5.12 Redis 50
5.13 AllegroGraph® RDFStore 52
5.14 Evaluatie 54

6 PERFORMANCETEST VAN DBMS-EN 55

6.1 Doelstelling en aanpak 55
6.2 Testsysteem 56
6.3 Benchmarktool 57
6.4 Omschrijving testdata 61
6.5 Selectie van de databasemanagementsystemen 64

6.6 OMSCHRIJVING TESTCASUSSEN 64
6.7 HANDLEIDING BENCHMARKTOOL: UITVOEREN VAN EEN TESTCASUS 71
6.8 TESTRESULTATEN VAN DE PERFORMANCETEST 72
7 CONCLUSIE EN AANBEVELINGEN 79
8 VERKLARENDE WOORDENLIJST 79
9 BIBLIOGRAFIE 79

1 Introductie

Het voor u liggende rapport beschrijft de aanleiding tot dit onderzoeksrapport en de bijbehorende onderzoeksresultaten. De aanpak die tot deze resultaten heeft geleid is beschreven in het *Onderzoeksplan*.

1.1 Inleiding

NCIM wil klanten uit de *Defensie & Veiligheid* sector kunnen demonstreren wat de mogelijkheden zijn van forensisch onderzoek op basis van sociale netwerken. Om dit doel te bereiken is een intern project gestart: het *Social Network Scanner* project (SNScanner). Tijdens het SNScanner project wordt een prototype ontwikkeld die de mogelijkheden van forensisch onderzoek op basis van sociale netwerken kan demonsteren. De SNScanner applicatie verzamelt openbare persoonsgegevens van Hyves, Twitter, Facebook en LinkedIn, legt onderlinge verbanden aan tussen deze data en slaat deze op in een database. Het huidige SNScanner prototype werkt met kleinere hoeveelheden testdata. In de toekomst moet de applicatie echter in staat zijn om grote hoeveelheden data te kunnen verwerken waarbij de snelheid van het beschikbaar stellen van deze grote hoeveelheden data ook van belang is.

Tijdens de ontwikkeling van het SNScanner prototype is er voor gekozen om de data op te slaan in een Derby database. Omdat er tijdens de ontwikkeling van het prototype geen tijd was om alternatieve databasemanagementsystemen (DBMS) te onderzoeken, is het onduidelijk of het gekozen DBMS de meest geschikte keuze is. Om dit probleem op te lossen, is een onderzoek uitgevoerd. Dit document beschrijft de resultaten van dit uitgevoerde onderzoek.

Het rapport begint met een korte introductie, de probleemstelling, de doelstelling en het startpunt. Hoofdstuk twee bevat de invulling van het onderzoek aan de hand van het opgestelde onderzoeksplan. Vervolgens worden de tijdens het literatuuronderzoek onderzochte datamodellen (hoofdstuk 3) en het DBMS gebruik van sociale netwerken (hoofdstuk 4) toegelicht. Ter afsluiting van het literatuuronderzoek zijn de eigenschappen van de DBMS-en, zoals gevonden tijdens 'DBMS gebruik bij sociale netwerken'-onderzoek en gerelateerde DBMS-en hiervan, op gestructureerde wijze gedocumenteerd (hoofdstuk 5). Hoofdstuk zes beschrijft de aanpak, de testopstelling – waaronder de benchmarktool - en de testresultaten van de performancetest. Tot slot eindigt het rapport met de conclusies en aanbevelingen.

1.2 Probleemstelling

Het is voor NCIM onduidelijk wat de geschiktheid is van de vele beschikbare DBMS-en met betrekking tot de opslag van grote hoeveelheden, onderling gekoppelde data.

1.3 Doelstelling

Het onderzoeken, testen en documenteren van geschikte DBMS-en voor de opslag van grote hoeveelheden data.

Om de doelstelling te bereiken wordt tijdens het onderzoek de volgende **hoofdvraag** beantwoord:

- Wat is het meest geschikte DBMS voor de massa opslag van onderling gekoppelde, sociale data?

Om de hoofdvraag te kunnen beantwoorden, worden de volgende **deelvragen** tijdens het onderzoek beantwoord:

1. Wat voor typen databasemanagementsystemen/datamodellen zijn er?
2. Voor welke doeleinden zijn deze typen het meest geschikt?
3. Wat zijn de nieuwste ontwikkelingen op het gebied van databasemanagementsystemen?
4. Welke databasemanagementsystemen worden gebruikt door sociale netwerken?
5. Voor welke doeleinden worden deze databasemanagementsystemen gebruikt?

1.4 Startpunt

Binnen NCIM zijn meerdere professionals gespecialiseerd in Oracle producten (1). Tot deze producten behoren ook het Oracle DBMS en MySQL. Verder is er onder andere kennis van MS SQL Server en het derby DBMS (door gebruik in het SNScanner project). Allen betreft het relationele databasemanagementsystemen.

2 Onderzoeksplan

Sectie 2.1 beschrijft de probleemanalyse van het Social Network Scanner project. Vervolgens wordt in sectie 2.2 de totstandkoming van het onderzoek beschreven. Sectie 2.3 beschrijft de planning tijdens het onderzoek. Tot slot beschrijven sectie 2.4 tot en met sectie 2.6 de invulling van het literatuuronderzoek, de ontwikkeling van de benchmarktool en de invulling van de performancetest.

2.1 Probleemanalyse: het Social Network Scanner project

Aanleiding voor het onderzoek is het missende onderzoek naar het meest geschikte databasemanagementsysteem voor de opslag van grote hoeveelheden data voor gebruik bij het Social Network Scanner (SNScanner) project. Dit hoofdstuk beschrijft de aanpak tijdens de analyse van het SNScanner project en de resultaten hiervan.

2.1.1 Analyseaanpak: het interviewen van projectleden

Omdat er bij de keuze van het databasemanagementsysteem geen onderzoek was gedaan bij het SNScanner project, was het onduidelijk wat het meest geschikte databasemanagementsysteem voor het SNScanner project is. Gezien deze onduidelijkheid was het nodig om eerst de requirements aan het DBMS te achterhalen, voordat er verder gegaan kon worden met het onderzoek. Omdat de projectleden van het SNScanner project het beste wisten - gezien het dagelijkse gebruik - waar het huidige DBMS tekortschoot, is besloten om de projectleden te interviewen om zo de requirements aan het DBMS te achterhalen. Om zoveel mogelijk informatie te achterhalen, is besloten de projectleden gezamenlijk te interviewen. Zo konden de projectleden elkaar aanvullen en de eigen opvattingen waar nodig verdedigen of bijstellen. Om het interview open te houden voor discussie, is er een beperkt aantal vragen opgesteld waardoor de richting van het interview bepaald werd door ‘klachten’ van de projectleden.

De volgende vragen zijn opgesteld en besproken tijdens het interview:

1. Welk databasemanagementsysteem wordt nu gebruikt?
2. Met wat voor soorten data heeft het SNScanner project te maken?
3. Met wat voor hoeveelheden data heeft het project te maken?
4. Welke tekortkomingen heeft het huidige databasemanagementsysteem?
5. Wat zijn volgens jullie belangrijke eisen aan het nieuwe databasemanagementsysteem?

2.1.2 Resultaat interview: criteria databasemanagementsysteem

Tijdens het interview kwam naar voren dat het huidige *Derby* databasemanagementsysteem niet voldeed aan de projecteisen. Het lezen van data uit de database was traag en het onderhouden van het relationele databaseschema kostte veel tijd, omdat de data tijdens de ontwikkeling qua structuur steeds veranderde. Aan de hand van het interview zijn de volgende requirements opgesteld:

1. Schaalbaarheid: Het SNScanner prototype werkt momenteel met kleine hoeveelheden testdata. In de toekomst moet de applicatie echter in staat zijn om grote hoeveelheden data te verwerken. Het databasemanagementsysteem moet mee kunnen schalen met de datagroei.

2. Snel lezen: Data moet te allen tijde snel beschikbaar zijn voor de gebruiker (bijvoorbeeld een politieonderzoeker) zodat onderzoek zo min mogelijk vertraging oploopt. Wanneer data gelijktijdig gelezen en geschreven wordt, mag het schrijven geen invloed hebben op de

leessnelheid. De snelheid van data schrijven heeft minder prioriteit dan het lezen.

3. Veel relaties: De gehele applicatie is gebaseerd op het ophalen van onderling gerelateerde data. De data in de database hebben onderling veel relaties. Relaties in de database moeten gemakkelijk aangelegd en onderhouden kunnen worden.

4. Dynamische data en uitbreidbaarheid: De data wordt opgehaald van meerdere bronnen en heeft geen vaste structuur. Het DBMS moet om kunnen gaan met dynamische data. Nieuwe databronnen (zoals een nieuw sociaal netwerk) moeten gemakkelijk toegevoegd kunnen worden.

5. Java ondersteuning: Het SNScanner prototype is een Java applicatie. Het DBMS moet vanuit Java applicaties benaderd kunnen worden.

Aan de hand van de SNScanner analyse is het verdere onderzoek ingevuld.

2.2 Totstandkoming van het onderzoek

Aan de hand van de gestelde criteria – achterhaald tijdens het interview – is de verdere invulling van het onderzoek bepaald. Deze sectie beschrijft de invulling van het uitgevoerde onderzoek.

2.2.1 Databasemanagementsysteem gebruik bij sociale netwerken

Om te kunnen bepalen welk databasemanagementsysteem het meest geschikt is voor gebruik in het SNScanner project, moesten de verschillende mogelijkheden onderzocht worden. Omdat er veel databasemanagementsystemen beschikbaar zijn, was het onmogelijk al deze DBMS-en te onderzoeken. Om te beginnen konden willekeurige DBMS-en onderzocht worden, maar dan was de kans groot dat dit had geleid tot resultaten die niet overeenkwamen met de gestelde requirements aan het databasemanagementsysteem voor het SNScanner project. Omdat het SNScanner project werkt met opgehaalde data van de sociale netwerken Hyves, Twitter, Facebook en LinkedIn, was het interessanter om te onderzoeken hoe deze websites zelf omgaan met de opslag van de vaak enorme hoeveelheid data. Wanneer de sociale netwerken allemaal gebruik zouden maken van een bepaald databasemanagementsysteem was deze informatie wél relevant voor gebruik bij de DBMS keuze voor het SNScanner project. Er is dan ook besloten het onderzoek te beginnen met het achterhalen van het DBMS gebruik bij sociale netwerken.

Zie hoofdstuk 4 voor meer informatie over het onderzoek naar het DBMS gebruik bij sociale netwerken.

2.2.2 Onderzoeken overige databasemanagementsystemen

NCIM gaf aan ook graag een algemeen overzicht van databasemanagementsystemen te verkrijgen. Het algemeen overzicht zou dan als hulpmiddel gebruikt kunnen worden bij de DBMS keuze voor toekomstige projecten. Om een algemeen overzicht te krijgen, is besloten naast het onderzoeken van de DBMS-en in gebruik door de sociale netwerken ook andere DBMS-en te onderzoeken. Zie hoofdstuk 5 hoe tot de selectie van deze overige databasemanagementsystemen is gekomen.

2.2.3 Performancetest met behulp van een benchmarktool

Omdat onder andere de leessnelheid van het nieuwe databasemanagementsysteem een gestelde requirement was, moest de performance van de verschillende onderzochte DBMS-en getest worden.

Er zijn meerdere tools beschikbaar om de performance van een DBMS te testen, maar gezien de afstudeerrichting van de afstudeerder (*systeemontwikkeling)* is ervoor gekozen een eigen benchmarktool te gaan ontwikkelen. De ontwikkeling van de benchmarktool was een belangrijk onderdeel van het onderzoek: met behulp van de benchmarktool is de performancetest uitgevoerd.

2.3 Planning

De planning is verdeeld aan de hand van de drie onderdelen tijdens het onderzoek:

1. Literatuuronderzoek naar geschikte DBMS-en.
2. Ontwikkelen benchmarktool.
3. Testen van DBMS-en die tijdens literatuuronderzoek als geschikt naar voren komen.

Onderstaande tabel geeft een overzicht van de werkzaamheden en resultaten tijdens de negen weken van het onderzoek.

Literatuuronderzoek

Week	Werkzaamheden	Resultaten	Begindatum	Einddatum
1	- Onderzoek typen DBMS-en - Onderzoek *sociale* DBMS-en - Ontwikkelen onderzoeksrapport	- Concept onderzoeksrapport	14-02-2011	18-02-2011
2	- Literatuuronderzoek - Ontwikkelen onderzoeksrapport	- Concept onderzoeksrapport	21-02-2011	25-02-2011
3	- Literatuuronderzoek - Ontwikkelen onderzoeksrapport	- Concept onderzoeksrapport	28-02-2011	04-03-2011

Ontwikkelen benchmarktool

Week	Werkzaamheden	Resultaten	Begindatum	Einddatum
4	- Ontwerpen benchmarktool - Ontwikkelen benchmarktool	- Concept benchmarktool	07-03-2011	11-03-2011
5	- Concept benchmarktool met opdrachtgever evalueren - Ontwerpen benchmarktool - Ontwikkelen benchmarktool	- Concept benchmarktool	14-03-2011	18-03-2011
6	- Ontwerpen benchmarktool - Ontwikkelen benchmarktool - Documenteren benchmarktool	- Benchmarktool	21-03-2011	25-03-2011

Testen databasemanagementsystemen

Week	Werkzaamheden	Resultaten	Begindatum	Einddatum
7	- Benchmarken DBMS-en - Ontwikkelen onderzoeksrapport	- Concept onderzoeksrapport	05-04-2011	08-04-2011
8	- Benchmarken DBMS-en - Ontwikkelen onderzoeksrapport		11-04-2011	15-04-2011
9	- Benchmarken DBMS-en - Ontwikkelen onderzoeksrapport	- Onderzoeksrapport - Benchmarktool met implementatie van onderzochte DBMS-en. - Beschrijving benchmarktool	18-04-2011	22-04-2011

Tabel 1 – Planning tijdens onderzoeksfase

2.4 Invulling literatuuronderzoek

Het literatuuronderzoek staat in het teken van het onderzoeken en documenteren van DBMS-en op basis van aanwezige digitale literatuur op het internet.

De doelstelling voor het literatuuronderzoek luidt:

- Het documenteren van de verschillende beschikbare DBMS-en.

2.4.1 Typen databasemanagementsystemen

Het literatuuronderzoek start met het documenteren van de verschillende typen DBMS-en. Zo zijn bijvoorbeeld onder meer relationele- en object georiënteerde DBMS-en van elkaar te onderscheiden. Tijdens de eerste week van het onderzoek worden deze en overige typen gedocumenteerd. Vragen die tijdens dit deelonderzoek worden beantwoord luiden:

1. Wat voor typen databasemanagementsystemen/datamodellen zijn er?
2. Voor welke doeleinden zijn deze typen het meest geschikt?
3. Wat zijn de nieuwste ontwikkelingen op het gebied van databasemanagementsystemen?

2.4.2 Databasemanagementsystemen bij sociale netwerken

Omdat de SNScanner applicatie met grote hoeveelheden data van sociale netwerken werkt, wordt er gestart met het onderzoeken en documenteren van de DBMS-en in gebruik door de sociale netwerken Twitter, Hyves, Facebook en LinkedIn. De volgende vragen worden tijdens dit deelonderzoek beantwoord:

1. Welke databasemanagementsystemen worden gebruikt door sociale netwerken?
2. Voor welke doeleinden worden deze databasemanagementsystemen gebruikt?

2.4.3 Overige databasemanagementsystemen

Het is mogelijk dat er tijdens het onderzoeken van de *sociale DBMS-en* informatie naar voren komt die de richting van de rest van het onderzoek bijstelt. Wanneer alle sociale netwerken bijvoorbeeld gebruik maken van een bepaald type DBMS kan het nuttig zijn om het onderzoek meer op dat soort typen DBMS-en te richten. De richting van het overige onderzoek wordt dus bepaald in week twee van het onderzoek (zie hoofdstuk *Planning*).

NCIM stelt een aantal randvoorwaarden aan een DBMS voor gebruik bij een project. Om tot een voor NCIM relevant onderzoeksrapport te komen, worden voor dit deel van het onderzoek alleen de DBMS-en onderzocht en gedocumenteerd die aan deze randvoorwaarden voldoen. De randvoorwaarden waaraan voldaan moet worden luiden:

1. Het DBMS biedt minimaal ondersteuning voor Windows en/of Linux.
2. Er is voldoende documentatie aanwezig betreft het installeren en de kenmerken van het DBMS.

Om voor alle DBMS-en tot een compleet en vergelijkbaar overzicht te komen wordt tijdens het literatuuronderzoek gebruik gemaakt van onderstaande checklist. De technische termen in de checklijsten zijn in het hoofdstuk *Verklarende woordenlijst* toegelicht. De in de checklist opgenomen kenmerken zijn gekozen in overleg met de opdrachtgever.

Kenmerken databasemanagementsysteem

Algemeen	
Naam	De naam van het databasemanagementsysteem.
Ontwikkelaar	De ontwikkelaar van de software.
Eerste release	Datum van eerste uitgave DBMS.
Laatste stable release	Datum van laatste stabiele uitgave.
Release frequentie	Hoe vaak een nieuwe stable release wordt uitgebracht.
Geschreven in	De taal waarin het DBMS is ontwikkeld (bijvoorbeeld Java of C).
Programmeertaal compatibiliteit	Voor welke programmeertalen het DBMS ondersteuning biedt.
Besturingssysteem compatibiliteit	De besturingssystemen waarop de databaseserver kan draaien.
Datamodel	Type datamodel, bijvoorbeeld object georiënteerd of relationeel.
Licentie	Licentie waaronder het DBMS is uitgebracht (bijvoorbeeld de Apache License).
Website	Officiële website van het DBMS.
Verbinding	
Embedded mode	Kan het DBMS in *embedded mode* draaien?
Client/server mode	Kan het DBMS in *client/server mode* draaien?
Querytaal	Door middel van welke taal kunnen er queries op het DBMS uitgevoerd worden?
JDBC driver	Biedt het DBMS support voor *JDBC*?
Stored procedures	Ondersteund het DBMS *Stored procedures?*
Distributie en betrouwbaarheid	
Automatic horizontal partitioning	Biedt het DBMS een vorm aan van automatische *Horizontal partitioning*?
Automatic vertical partitioning	Biedt het DBMS een vorm aan van automatische *Vertical partitioning*?
Automatic Sharding	Biedt het DBMS een vorm aan van automatische Sharding?
Replication	Biedt het DBMS een vorm aan van *Replication*?
Referential integrity	Biedt het DBMS een vorm aan van *Referential integrity*?
Transactions	Ondersteund het DBMS T*ransactions*?
Integrity model	Wat is het *Integrity model* van het DBMS?
Database	
Ondersteuning datatypen	Welke datatypen en groottes van datatypen worden ondersteund? Bijvoorbeeld text, integer en date.
Max. grootte db	De maximale grootte die de totale database kan aannemen.

Max. grootte db tabel	De maximale grootte die een enkele tabel in de database kan aannemen.
Max. grootte db rij	De maximale grootte die een enkele rij in de database kan aannemen.
Max. aantal kolommen per rij	Het maximaal aantal kolommen per rij.
Opmerkingen	
• Plaats voor extra commentaar.	
Meest geschikt voor	
• Toelichting voor welk gebruikt het DBMS geschikt is.	

2.5 Invulling benchmarktool ontwikkeling

Na het literatuuronderzoek wordt de benchmarktool ontwikkeld. Door middel van de benchmarktool wordt de performance van de geselecteerde DBMS-en getest. De doelstelling tijdens deze fase luidt:

- Het ontwikkelen en documenteren van een tool waarmee de performance van een DBMS getest kan worden.

Zie hoofdstuk vijf voor meer informatie over de testopstelling en de uit te voeren performancetests .

2.5.1 Programma van eisen

Deze sectie beschrijft de functionele- en niet-functionele eisen aan de benchmarktool. Eisen zijn aan de hand van de MoSCoW-methode geprioriteerd. De MoSCoW-methode verdeeld de eisen in onderstaande vier categorieën:

1. Must – deze eis moet in het eindresultaat terugkomen. Zonder deze eis is het eindresultaat onbruikbaar.
2. Should – zeer gewenste eis, maar zonder deze eis is het eindresultaat wel bruikbaar.
3. Could – aan deze eis wordt alleen voldaan wanneer er voldoende tijd is.
4. Won't – interessante eis voor de toekomst maar wordt tijdens dit project niet geïmplementeerd.

Must

1. Benchmarktool wordt ontwikkeld als Java applicatie.
2. Alle testcases moeten in dezelfde testopstelling uitgevoerd worden.
3. Alle testcases worden drie keer uitgevoerd om externe invloeden uit te uitsluiten.
4. De snelheid van schrijf- en leesacties van een DBMS kunnen meten.

Should

1. Gemeten testresultaten worden opgeslagen.
2. Complexe code wordt door middel van commentaar toegelicht.
3. De benchmarktool is uit te breiden voor toekomstig gebruik.
4. Er kan makkelijk geschakeld worden tussen de te meten DBMS-en.
5. Benchmarking van een DBMS kan via *command line* worden gestart.
6. Meerdere queries kunnen gelijktijdig uitgevoerd worden (*concurrency*).
7. De code van de benchmarktool voldoet aan de *Coding standards* en *Naming Conventions* zoals beschreven in de whitepaper *Java™ Coding Style Guide* door Achut Reddy, Sun Microsystems, Inc. 1998.

Could

1. Gemeten testresultaten worden in een database opgeslagen.
2. De opgeslagen testresultaten kunnen in een grafiek getoond worden.
3. Hoeveelheid testdata is aanpasbaar door middel van een configuratiebestand.
4. Er vind versiebeheer plaats van de testresultaten wanneer metingen meerdere malen worden uitgevoerd.

Won't

Geen won't eisen .

2.5.2 Globaal ontwerp

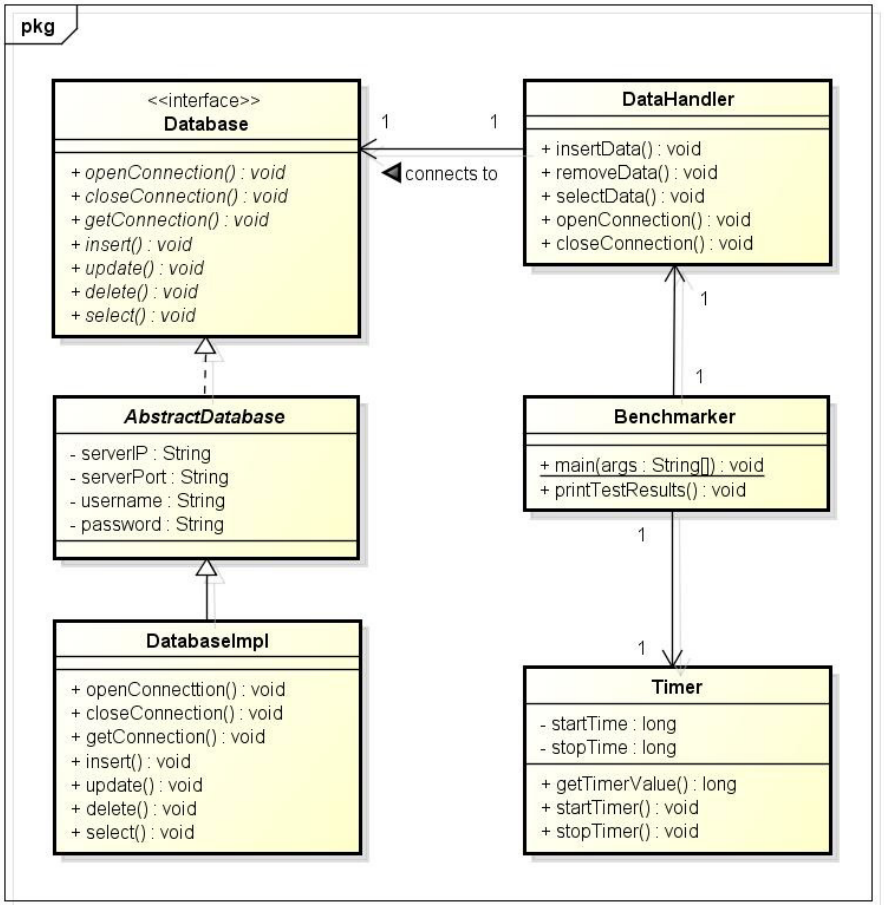
Figuur 1 toont het globale klassendiagram van de benchmarktool. Het ontwerp wordt tijdens de ontwikkeling van de benchmarktool uitgebreid.

De *Benchmarker* class is het centrale punt in de applicatie. Deze klasse zorgt voor de delegatie van de volgende functionaliteiten:

- het opzetten van de databaseconnectie (*DataHandler*);
- het starten, stoppen en verkrijgen van de benchmarktijd (*Timer*);
- en de delegatie tot het uitvoeren van queries op het DBMS (*DataHandler*).

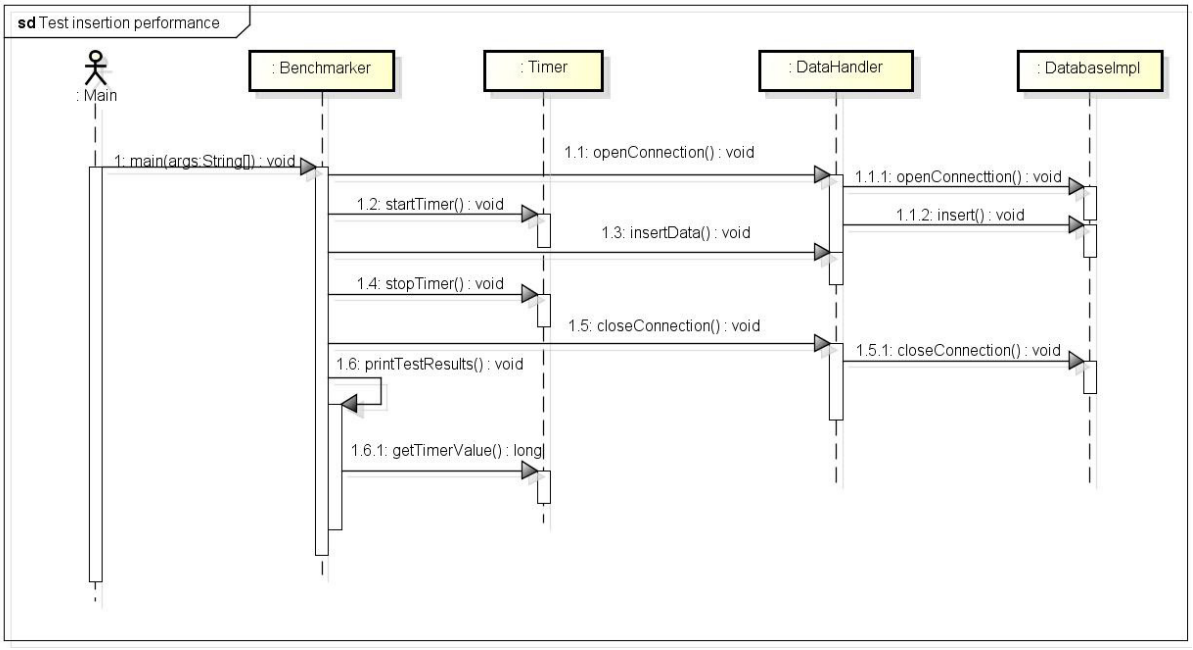
De database klasse is een interface van methoden die ingevuld moeten worden door de daadwerkelijke implementaties van de verschillende databases. Tussen de database interface en de database implementaties zit de abstracte *AbstractDatabase* klasse. Deze abstracte klasse verzorgt de implementatie van algemene database eigenschappen. De database implementaties verzorgen de implementatie van de specifieke databasemethoden.

De geïmplementeerde methodes beschrijven de specifieke zaken die per database verschillen. Elke database heeft bijvoorbeeld andere code voor het toevoegen (*insertion*) van data.



Figuur 1 - Globaal ontwerp benchmarktool

Onderstaand sequencediagram beschrijft de use case 'Het testen van de *insertion* performance':



Figuur 2 – Sequencediagram 'Het testen van de *insertion* performance'.

Zoals te zien in het sequencediagram start het programma met de aanroep van de main() methode in de Benchmarker klasse. Vervolgens delegeert de Benchmarker het openen van de databaseconnectie aan de DataHandler (*openConnection*). De DataHandler opent de connectie van een specifiek DBMS. Vervolgens wordt de Timer gestart (startTimer), de data in de database toegevoegd en wordt de Timer weer gestopt (*stopTimer)*. Tot slot wordt de databaseconnectie gesloten en de gemeten tijd geprint (*printTestResults*).

Om er voor te zorgen dat het openen en het sluiten van de databaseconnectie geen invloed hebben op de testresultaten, wordt de timer *na* het openen van de connectie gestart en *voor* het sluiten van de connectie gestopt.

2.6 Invulling performancetest

Dit hoofdstuk beschrijft de invulling van de uit te voeren performancetest. De performancetest wordt uitgevoerd met behulp van de ontwikkelde benchmarktool. De doelstelling van het uitvoeren van de performancetest luidt:

- De performance van de geselecteerde DBMS-en in verschillende situaties achterhalen met behulp van de benchmarktool.

De uiteindelijke testopstelling en geteste scenario's zijn beschreven in hoofdstuk 6.

2.6.1 Testsysteem

Voor het testsysteem wordt er gebruik gemaakt van hardware beschikbaar gesteld door de opdrachtgever. Het volgende testsysteem is ter beschikking gesteld voor het onderzoek:

Systeem	**Dell Latitude D830 X86 PC**
Processor	Intel™ Core™2 Duo CPU T7100 @ 1.80GHz, 1801 Mhz, 2 Core(s), 2 Logical Processor(s)
Total Physical Memory	2,00 GB
BIOS Version/Date	Dell Inc. A02, 7-6-2007
SMBIOS Version	2.4
Besturingssysteem	Microsoft Windows 7 Enterprise Version 6.1.7600 Build 7600

Om de performancetest op een later moment te kunnen reproduceren, dient er gebruik te worden gemaakt van een systeem met dezelfde eigenschappen. Er kan ook voor gekozen worden om alle tests op een ander systeem, bijvoorbeeld met een Linux besturingssysteem, te herhalen.

Om elk DBMS in een gelijkwaardige testomgeving te kunnen testen, wordt elke test met dezelfde, minimaal benodigde Windows processen uitgevoerd.

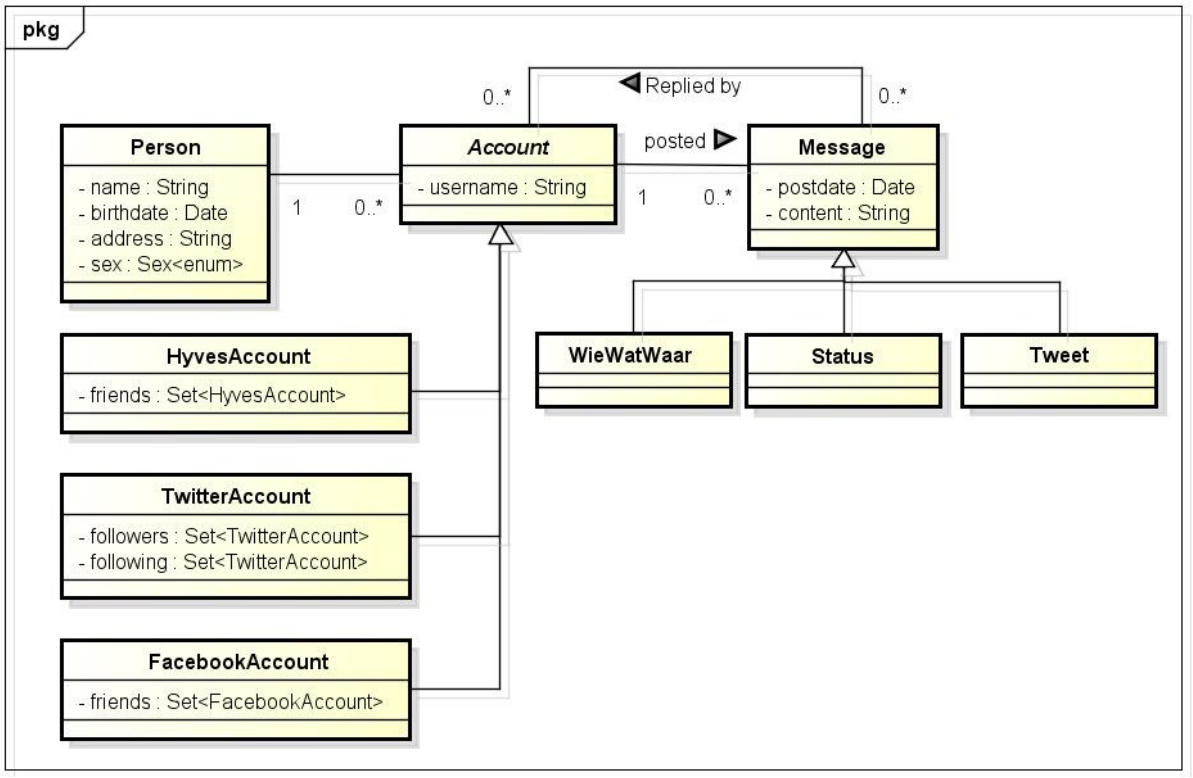
2.6.2 Testdata

De hoofdvraag die tijdens het onderzoek beantwoordt moet worden luidt:

> "Wat is het meest geschikte DBMS voor de massa opslag van onderling gekoppelde, sociale data?"

Om deze vraag, wat performance betreft te kunnen beantwoorden, wordt tijdens de performancetest gebruikt gemaakt van testdata, overeenkomend met het type data van sociale netwerken.

De testdata die we in de testopstelling gebruiken gaat uit van sterk versimpelde, onderling gekoppelde sociale data. Hieronder volgt een globaal klassendiagram van de testklassen. Tijdens de ontwikkeling van de benchmarktool wordt het ontwerp uitgebreid en wordt de hoeveelheid[1] testdata bepaald. Het definitieve ontwerp van de testklassen, zoals gebruikt tijdens de performancetest, wordt beschreven in het onderzoeksrapport.



Figuur 3 - Globaal ontwerp testklassen

In het ontwerp (Figuur 3) is een *Person*-klasse te zien. *Person* representeert een persoon met bijbehorende persoonsgegevens, zoals zijn geboortedatum, adres en geslacht. Een *Person* kan nul of meerdere *Accounts* hebben van verschillende sociale netwerken. Elk account heeft weer relaties (connecties) met andere accounts. Via elk *Account* kan de persoon openbare *Messages* sturen en op *Messages* van andere personen reageren.

[1] Hoeveelheid testdata: het aantal objecten van testklassen waarmee tijdens de performancetest gewerkt gaat worden. De database bevat bijvoorbeeld 1000 personen met elk twee accounts en gemiddeld 200 vrienden.

2.6.3 Aanpak

Bij het uitvoeren van de performancetest wordt gestart met het vastleggen van een aantal algemene zaken: de benodigde tijd om het DBMS op het testsysteem te installeren en de hoeveelheid benodigde tijd om de DBMS-functionaliteiten te implementeren in de benchmarktool.

Vervolgens wordt de daadwerkelijke performance van het DBMS getest op basis van de meest voorkomende queries in een applicatie: het toevoegen (insert), ophalen (select), veranderen (update) en verwijderen (delete) van data. Om de geschiktheid te testen van het DBMS met betrekking tot de hoeveelheid data waarmee gewerkt wordt, worden alle queries uitgevoerd op basis van drie grootte van datasets: kleine, middelgrote en grote datasets. De daadwerkelijke grootte van de datasets wordt tijdens de ontwikkeling van de benchmarktool bepaald zodat er gekeken kan worden wat het testsysteem aankan.

Het volledige overzicht van de te testen DBMS eigenschappen zijn weergegeven in Tabel 2.

Te testen eigenschappen

Installatietijd		De hoeveelheid benodigde tijd om het DBMS op het testsysteem te installeren.
Implementatietijd benchmarktool		De hoeveelheid benodigde tijd om de DBMS-functionaliteiten in de benchmarktool te implementeren.
Queries	Insert	Alle queries worden meerdere malen getest met verschillende hoeveelheden testdata. De performancetest wordt uitgevoerd met een kleine, middelgrote en grote dataset.
	Select	
	Update	
	Delete	
Concurrency		Om de performance van het DBMS te testen wanneer meerdere queries gelijktijdig (concurrency) worden uitgevoerd, worden alle testcases single- en multi-threaded uitgevoerd.

Tabel 2 – Te testen database eigenschappen

Om de *concurrency* performance van de DBMS-en te kunnen testen worden de testcases single- en multi-threaded uitgevoerd. Bij de single-threaded cases wordt één verzoek op het DBMS uitgevoerd. De multi-threaded cases zorgen voor meerdere, gelijktijdige verzoeken naar het DBMS.

Om externe invloeden uit te sluiten wordt elke test drie maal uitgevoerd. De performance van de DBMS-en worden aan de hand van het gemiddelde en de standaarddeviatie van de resultaten vergeleken.

2.6.4 Randvoorwaarden

Gezien de geringe duur van het onderzoek kunnen niet alle, tijdens literatuuronderzoek gedocumenteerde, DBMS-en getest worden. Er moeten keuzes gemaakt worden welke DBMS-en wel en welke DBMS-en niet getest worden. Om tot een selectie te komen van DBMS-en die getest gaan worden, komen alleen de DBMS-en in aanmerking die voldoen aan de volgende voorwaarden:

1. DBMS kan getest worden op het testsysteem.
2. Uit gedocumenteerde literatuuronderzoek is af te leiden dat het DBMS potentieel geschikt is voor de SNScanner applicatie.

2.6.5 Grenzen

Om de performancetest af te kunnen bakenen, worden er een aantal grenzen opgesteld. Deze sectie beschrijft de grenzen gesteld aan de performancetest.

DBMS optimalisaties

Voor de meeste DBMS-en zijn vele optimalisaties beschikbaar. Deze optimalisaties kunnen invloed hebben op de performance van het DBMS. Bij DBMS-en gaat het veelal om ingewikkeld uit te voeren optimalisaties en voor uitvoering van deze DBMS-optimalisaties is vaak uitgebreide kennis benodigd. Aan de hand van de beschikbare tijd en benodigde kennis voor uitvoering van de optimalisaties wordt bepaald welke optimalisaties in het onderzoek worden meegenomen. Om de performancetest eerlijk te laten verlopen, wordt voor elk DBMS evenveel tijd uitgetrokken voor de uitvoering van optimalisaties.

Application Programming Interface (API)

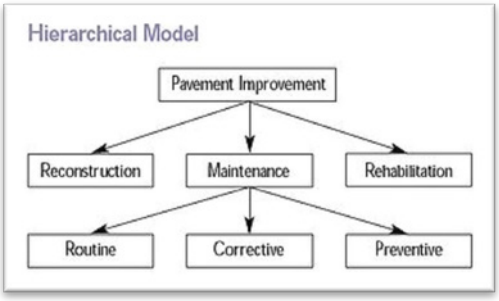
Voor Java zijn vele API's beschikbaar die gebruikt kunnen worden ter bevordering van de systeemontwikkeling. Dit zijn bijvoorbeeld bibliotheken die het gemakkelijker maken om Excel bestanden in te lezen. Ook zijn er API's die het uitvoeren van *queries* op het DBMS gemakkelijker maken. Tijdens het testen van de DBMS-en worden geen API's – buiten de standaard Java 1.6 SDK en door DBMS officieel meegeleverde API's - gebruikt die invloed hebben op de testresultaten.

3 Verschillende datamodellen

In welke structuur het databasemanagementsysteem de gegevens in een database opslaat, noemt men het datamodel. Dit hoofdstuk beschrijft de verschillende datamodellen en zijn bijbehorende kenmerken.

3.1 Het hiërarchisch model

Het hiërarchische datamodel (2) is een model waarbij de data wordt opgeslagen in een vorm, vergelijkbaar met die van een boomstructuur. Het hierarchische model bestaat uit *parent-child* relaties waarbij de gegevens van de *child* alleen via de *parent* bereikt kunnen worden. Het hierarchische model bestaat uit *een-op-veel* relaties waarbij de *parent* nul of meer *childs* heeft en een *child* altijd één *parent* heeft.

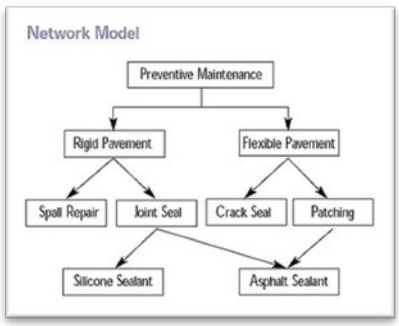


Figuur 4 – Het hiërarchisch model

Het hiërarchische model is ten tijde van de jaren ’50 en ’60 van de twintigste eeuw ontstaan en wordt hedendaags nauwelijks nog in productie gebruikt. De hiërarchische databases zijn in de praktijk vervangen door de *Netwerk databases*.

3.2 Het netwerk model

Het netwerk model (3) is de opvolger van het hiërarchische model. Het netwerk model bestaat, ten opzichte van het hierarchische een-op-veel relaties, uit *veel-op-veel* relaties waarbij *childs* meerdere *parents* kunnen hebben.

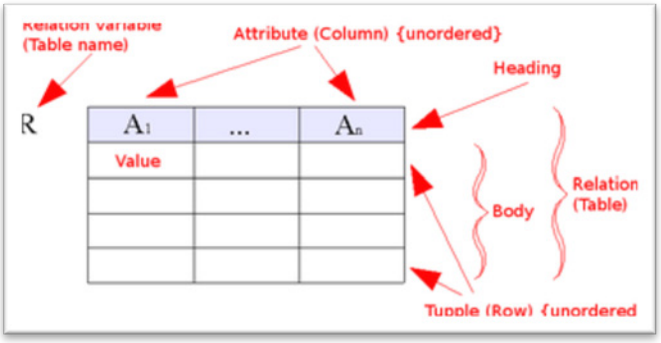


Figuur 5 – Het netwerk model

Sinds 1980 wordt het netwerk model in de praktijk vervangen door het relationele model.

3.3 Het relationele model

Het relationele datamodel (4) is opgebouwd volgens de regels van de *relationele algebra* en het wiskundige begrip *Relatie*.



Figuur 6 – Het relationele model

Een relatie in een relationele database is een tweedimensionale tabel, bestaande uit een *heading* en een *body* (Figuur 7). De *heading* is opgebouwd uit één of meer attributen (kolommen) en de *body* is een verzameling van *n tupels* (rijen). Een voorbeeld van een tabel in een relationele database is te zien in onderstaande tabel.

id	naam	geboortedatum	omschrijving
1	Ted Codd	23-08-1923	De grondlegger van het zeer invloedrijke relationele model voor databases.
2	Isaac Newton	04-01-1643	Had iets te maken met zwaartekracht?

Tabel 3 - Relationele databasetabel

In
Tabel 3 is een relatie 'Persoon' te zien waarin een id, naam, geboortedatum en omschrijving kan worden opgeslagen. De *attributen* in dit voorbeeld zijn *id, naam, geboortedatum* en *omschrijving*. In het volgende voorbeeld zijn twee tupels opgeslagen in de *Persoon*-tabel:

{1, Ted Codd, 23-08-1923, De grondlegger etc.}
{2, Isaac Newton, 04-01-1643, Had iets etc.}

Figuur 7 - Voorbeeld tupels RDBMS

Mutaties op de database – zoals het toevoegen, bewerken en verwijderen van gegevens – worden gedaan met behulp van de querytaal SQL (Structured Query Language).

> SQL (5): Een database-subtaal om gegevens in een relationele database op te vragen en bij te werken, en om de database te beheren. Hoewel het geen programmeertaal als C en Pascal is, kan SQL worden gebruikt om interactieve opvraagverzoeken (queries) te formuleren; SQL kan ook in een applicatie zijn opgenomen als instructie om gegevens te manipuleren. De SQL-standaard bevat ook onderdelen om gegevens te definiëren, wijzigen, besturen en beveiligen.

Er zijn vele relationele databasemanagementsystemen (RDBMS) beschikbaar. RDBMS-en worden hedendaags in de praktijk veelal gebruikt.

3.4 Object-oriented model

Een object-oriented database – ook wel object database - is een uitbreiding op de relationele database in die zin dat het werkt met *objecten* in plaats van *tupels*. De o*bjecten* in object databases komen overeen met objecten zoals die gebruikt worden in object georiënteerde programmeertalen.

Door data direct als object in de database op te slaan is er geen vertaalslag nodig bij het ophalen van diezelfde data. Als er bijvoorbeeld een object van het type *Persoon* opgeslagen wordt, kan dit object er weer als *Persoon* uitgehaald worden. Een ander voordeel voor object georiënteerde programmeurs is het consistente datamodel tussen de applicatie en database.

3.5 NoSQL

De term NoSQL werd geïntroduceerd in 1998 toen de term gebruikt werd als aanduiding voor een opensource, relationele database die geen gebruik maakte van de querytaal SQL (6). De term NoSQL werd in 2009 opnieuw in het leven geroepen: er werd een bijeenkomst georganiseerd met als doel een discussie over gedistribueerde databases. Sindsdien wordt met de term NoSQL databasemanagementsystemen aangeduid die niet voldoen aan de kenmerken van een klassiek relationeel DBMS. Voorbeelden van kenmerken waar NoSQL databasemanagementsystemen veelal afwijken van de traditionele relationele DBMS-en:

1. Geen vaste tabel schema's, er wordt geen structuur aan de data opgelegd.
2. Joins (zoals bij SQL) worden vermeden.
3. Horizontale schaalbaarheid.

Meer informatie over het ontstaan en concept van NoSQL is te vinden in de paper "NoSQL Databases" door Christof Strauch (6).

In onderstaande paragrafen worden de verschillende typen modellen toegelicht die worden geschaard onder NoSQL: de *key-value*, *column-oriented, document-oriented* en de *graph* databasemanagementsystemen.

3.5.1 Key-value

In de basis ondersteunen key-value systemen de opslag van een key/value combinatie. De waarde van de key is de unieke sleutel waarmee de gegevens geïdentificeerd kunnen worden – vergelijkbaar met de *primary key* bij het relationele model - en de value, de daadwerkelijke informatie. In plaats van de querytaal SQL, kennen key-value databases in de basis alleen *put, get* en *delete* commando's om mutaties op de database uit te voeren.

```
put("applicationVersion", "1.3")
get("applicationVersion") returns "1.3"
delete("applicationVersion")
get("applicationVersion") returns NULL
```

Figuur 8 – Voorbeeld mutatie key-value database

3.5.2 Column-oriented

Een column-oriented DBMS is een databasemanagementsysteem die de gegevens per kolom opslaat in plaats van per rij zoals bij een relationeel DBMS. De representatie van twee rijen ziet er in een column-oriented model als volgt uit:

```
{1, 2}
{Ted Codd, Isaac Newton}
{23-08-1923, 04-01-1643}
{De grondlegger van etc., Had iets met etc.}
```

Figuur 9 – Twee rijen in een column-oriented datamodel

Door de structuur van het column-oriented model zijn column-oriented systemen zeer efficiënt wanneer er van meerdere records (rijen) slechts een klein aantal kolommen gemuteerd dienen te worden.

3.5.3 Document-oriented

Het grote verschil tussen een RDBMS en een *document-oriented* DBMS is de flexibiliteit van de structuur van de data. Bij relationele databases wordt op voorhand de structuur van een tabel vastgelegd: bij het voorbeeld in Tabel 3 moet elke rij bestaan uit een *naam, geboortedatum* en *omschrijving*. Wanneer een rij daar niet aan voldoet – en dit niet door het DBMS wordt afgedwongen - wordt er in de lege kolom de waarde *NULL* opgeslagen. Een document-oriented DBMS heeft hier geen last van, de opgegeven data bepaald de structuur. De ene persoon kan bijvoorbeeld alleen uit een naam bestaan, terwijl een ander persoon alleen een geboortedatum heeft. Het volgende voorbeeld toont de flexibiliteit van een document-oriented DBMS:

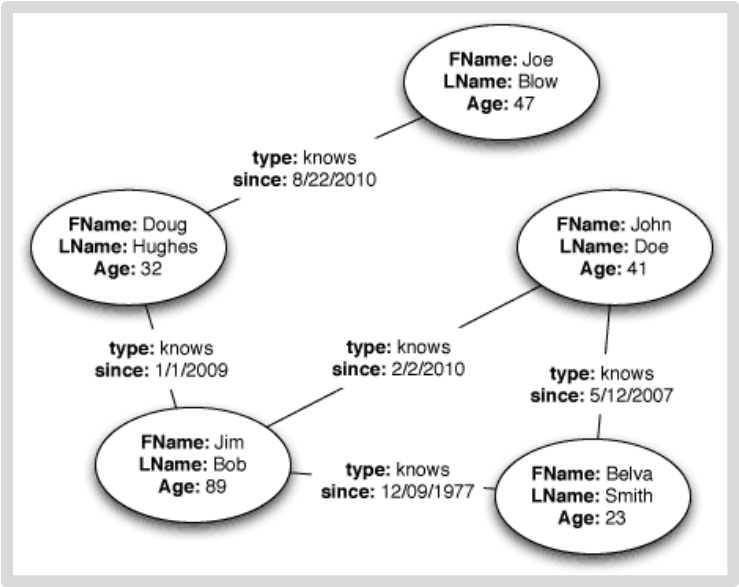
```
{
   “id” : 1,
   “name” : “Ted Codd”
}, {
   “id” : 2,
   “geboortedatum” : “04-01-1643”
}
```

Figuur 10 – Het schema vrije document-oriented datamodel

3.5.4 Graph (Graaf)

Graph databases zijn gebaseerd op de grafentheorie uit de wiskunde. Een graph database bestaat uit *nodes*, *properties* en *edges* waarbij de *nodes* te vergelijken zijn met objecten zoals bekend bij object-georiënteerde programmeertalen. *Properties* beschrijven de informatie van een *node* en *edges* zijn de onderlinge verbindingen tussen de *nodes*. Figuur 11 toont een voorbeeld van een graaf waarin relaties (*knows*) tussen personen zijn vastgelegd.

In het voorbeeld zijn de personen de *nodes.* De eigenschappen van de personen en de beschrijvingen van de verbindingen tussen de personen zijn de *properties*. En tot slot worden de verbindingen tussen de *nodes* de *edges* genoemd.



Figuur 11 – Representatie van een graaf

4 Het DBMS gebruik bij de sociale netwerken

Het *Social Network Scanner* project werkt met opgehaalde data van de sociale netwerken Hyves, Twitter, Facebook en LinkedIn. Door het gebruik van soortgelijke data in het SNScanner project was het interessant om te onderzoeken hoe deze websites zelf omgaan met de opslag van de, vaak grote hoeveelheden, data. Dit hoofdstuk beschrijft de doelstelling, de aanpak en de resultaten van dit onderzoek.

4.1 Doelstelling en Aanpak

De doelstelling tijdens dit deel van het onderzoek is het achterhalen van het DBMS gebruik bij Hyves, Twitter, Facebook en LinkedIn. De volgende vragen zijn gesteld tijdens het onderzoeken van elk sociaal netwerk:

1. Welke databasemanagementsystemen worden gebruikt?
2. Voor welke doeleinden worden deze databasemanagementsystemen gebruikt?

Om deze vragen te kunnen beantwoorden, moest er onderzocht worden waar en of deze informatie te vinden was. Er is gestart met brainstormen naar mogelijke bronnen van deze informatie. Hieruit kwamen de volgende oplossingen: informatie uit boeken, internetliteratuur en bedrijfslezingen op technische evenementen.

Er is begonnen met het zoeken van boeken op Amazon.com en Bol.com, omdat dit respectievelijk de grootste 'internetbibliotheken' zijn in Amerika en Nederland. Hoewel er genoeg verwante boeken vindbaar waren over bijvoorbeeld de Twitter API, was er geen boek vindbaar met informatie over het DBMS gebruik bij de sociale netwerken.

Vervolgens zijn met behulp van Google overige internetbronnen onderzocht. Tijdens het zoeken naar informatie op het internet, werd al snel duidelijk dat Twitter, Facebook en LinkedIn geen geheim maakte van hun DBMS gebruik. Twitter, Facebook en LinkedIn onderhielden allen een technisch blog waarin men berichten plaatsten over de nieuwste ontwikkelingen betreft de architectuur van het sociale netwerk, waaronder het DBMS gebruik. Technische informatie over Hyves was daarentegen onvindbaar.

Er worden veel technische evenementen georganiseerd, waar bedrijven tijdens lezingen een boekje open doen over de strategie en technische oplossingen van hun producten. Er is onderzocht of de sociale netwerken deelgenomen hebben aan deze lezingen zodat technische informatie achterhaald kon worden. Veel evenementen plaatsen de lezingen na afloop op hun website, waardoor eventuele bruikbare lezingen gemakkelijk vindbaar zijn. Voor de lezingen gold hetzelfde, waar veel informatie beschikbaar was over Twitter, Facebook en LinkedIn was Hyves wederom onvindbaar.

Technische informatie over Hyves was dus onvindbaar op het internet en daarom is er contact opgenomen met de technische e-mailsupport afdeling van Hyves. Op de gestelde vragen in de e-mail werd echter niet gereageerd. De gevonden informatie over Twitter, Facebook en LinkedIn was echter voldoende om een volledig onderzoek uit te kunnen voeren en daarom is besloten om Hyves niet verder te gaan onderzoeken. Omdat er genoeg informatie beschikbaar was via de officiële websites van de sociale netwerken, was het geen probleem dat er geen relevante boeken zijn gevonden.

4.2 Resultaten

Onderstaande sectie beschrijft de resultaten – het DBMS gebruik van Twitter, Facebook en LinkedIn - van het uitgevoerde literatuuronderzoek. Om te kunnen evalueren wat voor soort DBMS-en gebruikt worden is bij elk gevonden databasemanagementsysteem het type datamodel vermeld. Zie hoofdstuk 3 voor een toelichting van alle, in dit hoofdstuk vernoemde datamodellen.

4.2.1 Twitter

Twitter is begonnen met één relationeel MySQL databasemanagementsysteem. Maar sinds de start onder andere het aantal *tweets* (berichten) die Twitter dagelijks verwerkt enorm gestegen. Terwijl er in 2007 nog 5.000 tweets per dag werden verstuurd, was dit aantal in 2010 gestegen tot 50 miljoen tweets per dag (7).

MySQL was niet geschikt voor deze groei: de kosten die deze groei met zich meedroegen om de performance te waarborgen waren zo hoog – in termen van onderhoud – dat er gezocht moest worden naar een architectuur die op meer automatische wijze kon groeien.

Twitter heeft besloten om over te gaan op onderstaande oplossingen.

FlockDB - Om te achterhalen wie de volgers (followers) zijn van een bepaald persoon en wie deze persoon volgt (following), gebruikt Twitter het eigen ontwikkelde FlockDB (8).

Datamodel: Graph

HBase - Twitter gebruikt HBase voor het zoeken naar Twitter gebruikers (*People search*) (9).

Datamodel: Column-oriented

MySQL – MySQL wordt gebruikt voor de opslag van alle tweets en als basis voor FlockDB (8) (10).

Datamodel: Relationeel

Cassandra – Cassandra wordt gebruikt om de resultaten van datamining[2] over alle databases in op te slaan. De resultaten worden op de website gebruikt voor het tonen van de lokale trends en de populairste tweets (10).

Datamodel: Column-oriented

[2] Datamining is het gericht zoeken naar (statistische) verbanden in grote verzamelingen gegevens voor wetenschappelijke of commerciële doeleinden.

4.2.2 Facebook

Waar het aantal gebruikers in 2010 nog ongeveer 300 miljoen gebruikers betrof, is dit aantal het laatste jaar flink gestegen naar meer dan 500 miljoen actieve gebruikers (11). Met dit aantal actieve gebruikers mag Facebook zich met recht één van de grootste en bekendste sociale netwerk noemen. Om de groei te kunnen bijbenen en de beschikbaarheid van de dienst te kunnen waarborgen, zoekt Facebook actief naar betere oplossingen voor hun technische architectuur. Onderstaand een overzicht van het gebruik van databasemanagementsystemen bij Facebook ten tijde van het onderzoek:

HBase - Eind 2010 werden er vijftien biljoen berichten tussen Facebook gebruikers per maand verstuurd. Deze berichten werden door meer dan 350 miljoen verschillende gebruikers geproduceerd. De Facebook chat service zorgde nog eens voor 120 biljoen berichten per maand. De door Facebook gemeten gebruikspatronen toonden aan dat het om tijdelijke data ging (zoals chatberichten) die nauwelijks meerdere malen werden benaderd.
Als opslagoplossing voor deze diensten is HBase gekozen (12). Alternatieven waren MySQL en Cassandra. MySQL is ongeschikt bevonden, doordat de performance daalde naarmate de dataset groeide. Cassandra voldeed niet gezien het *eventual consistent* model (meer informatie: (13)). HBase toonde goede performance en schaalbaarheid, had een geschikter consistentiemodel en ondersteunde gestelde requirements zoals *load balancing* en *replication*.

Datamodel: Column-oriented

MySQL – MySQL is nog op grote schaal in gebruik door Facebook (14). Facebook gebruikt een *sharded* MySQL oplossing met InnoDB als storage engine voor de opslag van key/value data. De veelal trage join queries worden vermeden.

Datamodel: Relationeel

Cassandra - Toen Facebook zocht naar een oplossing voor de opslag van privé berichten die onderling tussen de vele Facebook gebruikers werden verstuurd, kwam men tot de conclusie (15) dat de beschikbare traditionele oplossingen niet geschikt waren. Daaropvolgend is Facebook gestart met de ontwikkeling van een eigen oplossing: Cassandra. Cassandra is een gedistribueerde, *column-oriented* database geschikt voor het beheer van grote hoeveelheden gestructureerde data verspreid over verschillende servers.

Datamodel: Column-oriented

4.2.3 LinkedIn

LinkedIn is gestart met één Oracle database. De oplossing was echter ongeschikt toen de hoeveelheid data en het aantal schrijfacties grotere vormen aannam (16).

Als oplossing heeft LinkedIn een combinatie van MySQL, Oracle en Voldemort bovenop het *Hadoop* File System. De door gebruikers en LinkedIn services gecreëerde data wordt direct in een online database - MySQL, Oracle of Voldemort - opgeslagen en geplaatst in een Hadoop datawarehouse. Vervolgens wordt deze data geanalyseerd en gestructureerd met behulp van verschillende offline processen en tot slot weer teruggeplaatst naar de live databases (Voldemort). Door de data offline te analyseren en structureren, ondervindt de gebruiker geen hinder van deze processen.

Voldemort – Een door LinkedIn ontwikkeld databasemanagementsysteem, specifiek ontworpen voor situaties waarbij snel, schaalbare lees- en schrijfacties nodig zijn.

Datamodel: Column-oriented

Oracle database – Oracle is al sinds de start van het netwerk in gebruik door LinkedIn. Het hedendaagse, specifieke gebruik van het DBMS is echter onbekend.

Datamodel: Object-relationeel

MySQL – door LinkedIn in gebruik voor levering van *read only* data. Er is gekozen voor de MyISAM storage engine (dit in tegenstelling tot InnoDB bij Facebook). Doordat MyISAM geen *ACID* transacties biedt - de InnoDB storage engine verzorgt dit wel - is MyISAM sneller voor *read only* data.

Datamodel: Relationeel

4.3 Evaluatie

De sociale netwerken hebben elk zijn eigen ideeën en oplossingen betreft de dataopslag. Wel zijn er opvallende overeenkomsten tussen de netwerken. Zo maken Twitter, Facebook en LinkedIn allen gebruik van het relationele databasemanagementsysteem MySQL en zijn allen deels overgestapt op *NoSQL* oplossingen (zie sectie 3.5) waarbij vooral de *column-oriented* DBMS-en HBase en Cassandra de voorkeur hebben.

Opvallend is dat alle sociale netwerken gebruik maken van een zelf ontwikkelde oplossing: FlockDB van Twitter, Cassandra van Facebook en Voldemort van LinkedIn. Een verklaring hiervoor kan zijn, dat voor de oprichting van de sociale netwerken (rond 2003-2004) geen enkele website te maken kreeg met de hoeveelheid data en gelijktijdige bezoekers waar de sociale websites hedendaags mee te maken hebben. De sociale netwerken kregen als één van de eerste websites (afgezien van de zoekmachines) te maken met deze grote hoeveelheden data en omdat de relationele databasemanagementsystemen niet voldeden, zijn er eigen oplossingen ontwikkeld.

Nu het duidelijk is welke databasemanagementsystemen gebruikt worden door Twitter, Facebook en LinkedIn is er echter nog geen gedetailleerde informatie over deze databasemanagementsystemen achterhaald. Het is dus nog onduidelijk of deze DBMS-en aan de gestelde criteria van het SNScanner

project voldoen. Tijdens het verdere onderzoek (volgende hoofdstuk) is gedetailleerde informatie over de gevonden databasemanagementsystemen achterhaald.

5 Overzicht databasemanagementsystemen

Dit hoofdstuk geeft een overzicht van de gevonden resultaten tijdens het onderzoeken van de verschillende databasemanagementsystemen. Om gestructureerd te werk te gaan en voor alle DBMS-en tot een compleet en vergelijkbaar overzicht te komen is voor elk DBMS dezelfde checklist van kenmerken – zoals opgesteld in het onderzoeksplan (sectie 2.4.3) - gebruikt.

5.1 Derby

Derby is in dit onderzoek opgenomen omdat het gebruikt wordt in het *Social Network Scanner* (SNScanner) project van NCIM. NCIM wil weten of Derby de juiste keuze is voor het SNScanner project.

Kenmerken

Algemeen	
Naam	Apache Derby
Ontwikkelaar	Apache Software Foundation
Eerste release	1997, destijds door Cloudscape Inc. uitgebracht als JBMS
Laatste stable release	Versie 10.7.1.1 (14-12-2010)
Release frequentie	Ongeveer drie keer per jaar nieuwe versie met zowel verbeteringen als nieuwe features.
Geschreven in	Java
Programmeertaal compatibiliteit	Java, PHP, .NET, Perl, PHP en overige talen die via DRDA protocol, JDBC en ODVC/CLI kunnen communiceren. (17)
Besturingssysteem compatibiliteit	Windows, Linux, UNIX en andere Java Virtual Machine (JVM) ondersteunende besturingssystemen. (18)
Datamodel	Relationeel
Licentie	Apache License 2.0
Website	http://db.apache.org/derby/
Verbinding	
Embedded mode	✔
Client/server mode	✔
Querytaal	SQL
JDBC driver	✔
Stored procedures	✔ (19)

Distributie en betrouwbaarheid	
Automatic horizontal partitioning	✗
Automatic vertical partitioning	✗
Automatic Sharding	✗
Replication	✓ (20)
Referential integrity	✓ (21)
Transactions	✓ (22)
Integrity model	ACID (22)
Database	
Ondersteuning datatypen (23)	BIGINT, BLOB, BOOLEAN, CHAR, CHAR FOR BIT DATA, CLOB, DATE, DECIMAL, DOUBLE, DOUBLE, PRECISION, FLOAT, INTEGER, LONG VARCHAR, LONG VARCHAR FOR BIT DATA, NUMERIC, REAL, SMALLINT, TIME, TIMESTAMP, VARCHAR, VARCHAR FOR BIT DATA, XML
Max. grootte db	Onbeperkt (24)
Max. grootte db tabel	Onbeperkt (24)
Max. grootte db rij	Onbeperkt (24)
Max. aantal kolommen per rij	1,012 kolommen (19)
Opmerkingen	
• Veel officiële en externe documentatie beschikbaar. • Bewezen product in de markt.	
Meest geschikt voor	
• Omgevingen waarbij de schaalbaarheid van data geen vereiste is. • Applicaties waarbij data integriteit (ACID) een vereiste is. • Applicaties waarbij relaties tussen verschillende data bestaan.	

5.1.1 Conclusie

Derby is een relationeel databasemanagementsysteem en ondersteund vele features. Echter wanneer schaalbaarheid van de data van belang is, is Derby geen geschikte keuze. Derby biedt namelijk geen ondersteuning voor *partitioning* en *sharding*.

5.2 MySQL

MySQL is inmiddels een bewezen, veelgebruikte dataopslag oplossing en is in gebruik door Facebook, Twitter en LinkedIn. Door het vele gebruik door de sociale netwerken is MySQL in dit onderzoek opgenomen.

MySQL biedt ondersteuning voor meerdere *storage engines*. Sinds versie 5.5 kiest MySQL InnoDB als standaard gebruikte storage engine (25). Het volgende MySQL overzicht is gedocumenteerd op basis van de opensource MySQL Community Edition met InnoDB als storage engine.

Kenmerken

Algemeen	
Naam	MySQL
Ontwikkelaar	MySQL AB
Eerste release	23 mei 1995
Laatste stable release	Versie 5.5.9 (07-02-2011)
Release frequentie	Hoog, meerdere releases per jaar.
Geschreven in	C en C++
Programmeertaal compatibiliteit	Onder andere C, Java, .NET en PHP.
Besturingssysteem compatibiliteit	Cross-platform (FreeBSD, HP-UX, Linux, Mac OS X, Solaris, Windows XP/Vista/Server 2003/Server 2008) (26)
Datamodel	Relationeel
Licentie	GPLv2
Website	www.mysql.com
Verbinding	
Embedded mode	✔ (MySQL Connector/MXJ)
Client/server mode	✔
Querytaal	SQL
JDBC driver	✔ (MySQL Connector/J)
Stored procedures	✔
Distributie en betrouwbaarheid	
Automatic horizontal partitioning	✔ (27)
Automatic vertical partitioning	✘ (27)
Automatic Sharding	✘ Niet zonder het gebruik van externe producten.
Replication	✔ Voor backup en verhoging van de beschikbaarheid (28).
Referential integrity	✔ (25)
Transactions	✔ (25)

Integrity model	ACID transactions met InnoDB als storage engine (25).
Database	
Ondersteuning datatypen (29)	BIT, TINYINT, BOOLEAN, SMALLINT, MEDIUMINT, INT, BIGINT, FLOAT, DOUBLE, FLOAT, DECIMAL, DATE, DATETIME, TIMESTAMP, TIME, YEAR, CHAR, VARCHAR, BINARY, VARBINARY, TINYBLOB, TINITEXT, BLOB, TEKST, MEDIUMBLOB, MEDIUMTEXT, LONGBLOB, LONGTEXT, ENUM, SET
Max. grootte db	Afhankelijk van besturingssysteem (30). Win32 met NTFS 2TB, Linux 2.4+ met ext3 4TB.
Max. grootte db tabel	64 TB (31) maar wel afhankelijk van besturingssysteem (30).
Max. grootte db rij	~ 8000 bytes exclusief Varbinary, Varchar, Blob en Text kolommen (32).
Max. aantal kolommen per rij	1000 kolommen (31).
Opmerkingen	
• Veel officiële en externe documentatie beschikbaar. • Bewezen product in de markt. • Wanneer er meer gelezen (*read*) wordt dan geschreven (*write*: *insert/update*), kan er beter gekozen worden voor MyISAM als storage engine. MyISAM is sneller omdat transacties niet ACID zijn.	
Meest geschikt voor	
• Applicaties waarbij data integriteit (ACID) een vereiste is. • Applicaties waarbij het wenselijk is om met opensource software te werken. • Applicaties waarbij relaties tussen verschillende data bestaan.	

5.2.1 Conclusie

MySQL is een relationeel databasemanagementsysteem met ondersteuning voor vele features. MySQL is inmiddels een bewezen product in de markt, is veel documentatie van aanwezig en is volledig opensource. Dit maakt MySQL tot een goede keuze voor gebruik bij applicaties waarbij een bewezen product van belang is. MySQL biedt echter geen officiële ondersteuning voor *vertical partitioning* en *sharding* waardoor de schaalbaarheid opties beperkt zijn.

5.3 HBase

HBase is een DBMS bovenop het *Hadoop Distributed File System*[3]. HBase is in gebruik door Twitter en Facebook en voor die reden opgenomen in dit onderzoek.

Kenmerken

Algemeen	
Naam	Apache HBase
Ontwikkelaar	Apache Software Foundation
Eerste release	2 februari 2008
Laatste stable release	Versie 0.90.1 (17-02-2011)
Release frequentie	Hoog, ongeveer zeven keer per jaar.
Geschreven in	Java
Programmeertaal compatibiliteit	Java via officiële API + programmeertalen die via de HBase gateway API's (zie *Querytaal*) kunnen communiceren.
Besturingssysteem compatibiliteit	Besturingssystemen met ondersteuning voor Java waarbij hoofdzakelijk gericht op *nix omgevingen zoals Linux en UNIX.
Datamodel	Column-oriented gebaseerd op Google's BigTable (33).
Licentie	Apache License 2.0
Website	http://hbase.apache.org
Verbinding	
Embedded mode	✗
Client/server mode	✓
Querytaal	API, REST, XML en SQL dialect via Hadoop HBql
JDBC driver	✓ Via externe API (34).
Stored procedures	✗ Wel *major* prioriteit [28].
Distributie en betrouwbaarheid	
Automatic horizontal partitioning	✓ via HDFS.
Automatic vertical partitioning	✓ via HDFS.
Automatic Sharding	✓ via HDFS.
Replication	✓ Voor backup en verhoging van de beschikbaarheid (35).
Referential integrity	✗

[3] "Hadoop Distributed File System (HDFS) is the primary storage system used by Hadoop applications. HDFS creates multiple replicas of data blocks and distributes them on compute nodes throughout a cluster to enable reliable, extremely rapid computations." - http://hadoop.apache.org/hdfs/

Transactions	✗
Integrity model	Gedeeltelijk ACID (36).
Database	
Ondersteuning datatypen (33)	Ondersteuning voor LONG (voor *timestamps*) en Byte arrays (voor *key/value* paren) (33).
Max. grootte db	Voor zover bekend, geen beperking vanuit HBase architectuur.
Max. grootte db tabel	
Max. grootte db rij	
Max. aantal kolommen per rij	
Opmerkingen	
• Relatief jong product. • Relatief weinig documentatie / kleine community. • Documentatie is erg technisch, hoge *learning curve*. • Documentatie is – soms onlogisch - verspreid over meerdere websites. • Voor gedistribueerde omgeving is HDFS kennis benodigd (37).	
Meest geschikt voor	
• Verwerking van grote hoeveelheden data. • Toepassingen waarbij data integriteit van ondergeschikt belang is. • Gedistribueerde real time toepassingen. • Logging van data.	

5.3.1 Conclusie

HBase is een column-oriented DBMS en richt zich vooral op gedistribueerde omgevingen waarbij de integriteit van de data van ondergeschikt belang is. Volgens de CAP-theorie is HBase een *Consistent, Partition-Tolerant* systeem wat het DBMS geschikt maakt voor toepassingen waarbij grote hoeveelheden data gepartitioneerd opgeslagen worden op verschillende servers en clients altijd met dezelfde versie (consistent) van de data werken. Het DBMS is echter minder geschikt voor toepassingen waarbij veel gelijktijdige schrijf- en leesacties plaatsvinden.

5.4 Apache Cassandra

Cassandra is door zowel Facebook als Twitter in gebruik. Dit gegeven maakt het interessant om het DBMS op te nemen in het onderzoek.

Kenmerken

Algemeen	
Naam	Apache Cassandra
Ontwikkelaar	Apache Software Foundation
Eerste release	Eerste release in 2008 opensource uitgebracht door Facebook (15) (38).
Laatste stable release	Versie 0.7.2 (16-02-2011)
Release frequentie	Hoog
Geschreven in	Java
Programmeertaal compatibiliteit	Pyton, Java, Grails, .NET, Ruby, PHP
Besturingssysteem compatibiliteit	Windows, Linux, UNIX en andere Java Virtual Machine (JVM) ondersteunende besturingssystemen.
Datamodel	Column-oriented gebaseerd op Googles BigTable (39) en Amazons Dynamo (40).
Licentie	Apache License 2.0
Website	http://cassandra.apache.org/
Verbinding	
Embedded mode	✗
Client/server mode	✓
Querytaal	Apache Thrift[4]
JDBC driver	✗
Stored procedures	✗
Distributie en betrouwbaarheid	
Automatic horizontal partitioning	✓ Door middel van *Consistent Hashing*. Gebaseerd op Amazon Dynamo (41) (40).
Automatic vertical partitioning	
Automatic Sharding	
Replication	✓ Gebaseerd op Amazon Dynamo (41) (40). (Zie Dynamo whitepaper - *4.3 Replication*)

[4] "Thrift is a software framework for scalable cross-language services development. It combines a software stack with a code generation engine to build services that work efficiently and seamlessly between C++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C#, Cocoa, Smalltalk, and OCaml." - http://incubator.apache.org/thrift/

Referential integrity	✗ Er zijn geen *secondary indexes* zoals *foreign keys*.
Transactions	✗
Integrity model	BASE
Database	
Ondersteuning datatypen	Ondersteuning voor LONG (voor *timestamps*) en Byte arrays (voor *key/value* paren).
Max. grootte db	Afhankelijk van *filesystem*, onbeperkte grootte met XFS filesystem op 64-bit systeem (42).
Max. grootte db tabel	Enige restrictie is dat data van één rij op de harde schijf past op een enkele node (machine). Data van één rij kan dus niet gedistribueerd worden over meerdere nodes (43).
Max. grootte db rij	
Max. aantal kolommen per rij	
Opmerkingen	
• Relatief jong product. • Relatief weinig documentatie / kleine community.	
Meest geschikt voor	
• Verwerking van grote hoeveelheden data. • Toepassingen waarbij ondersteuning van gemakkelijke, geavanceerde queries van ondergeschikt belang zijn ten opzichte van de performance. • Toepassingen met meer schrijf dan leesacties.	

5.4.1 Conclusie

Cassandra is een column-oriented DBMS, gebaseerd op Google's BigTable en is volgens de CAP-theorie een *Available, Partition-Tolerant* DBMS. Dit gegeven maakt het DBMS geschikt voor omgevingen waarbij gelijktijdige schrijf- en leesacties plaatsvinden. Het DBMS is echter minder geschikt voor toepassingen waarbij de integriteit en consistentie van data van belang zijn.

5.5 Project Voldemort

Voldemort is een door LinkedIn ontwikkeld DBMS en logischerwijs ook in gebruik door dit sociale netwerk. Gezien het gebruik van het DBMS bij LinkedIn is het DBMS in dit onderzoek opgenomen.

Kenmerken

Algemeen	
Naam	Project Voldemort
Ontwikkelaar	SNA LinkedIn[5]
Eerste release	Versie 0.51 (06-06-2009)
Laatste stable release	Versie 0.81 (17-06-2010)
Release frequentie	In het laatste jaar zeven releases.
Geschreven in	Java
Programmeertaal compatibiliteit	Java en gedeeltelijke support voor C gebaseerde talen (Python, PHP, Ruby, Perl, etc.) (44).
Besturingssysteem compatibiliteit	Windows, Linux, UNIX en andere Java Virtual Machine (JVM) ondersteunende besturingssystemen.
Datamodel	Key-Value model gebaseerd op Amazons Dynamo (45).
Licentie	Apache 2.0 license
Website	http://project-voldemort.com/
Verbinding	
Embedded mode	✔ (46)
Client/server mode	✔
Querytaal	Eigen API en Thrift[4].
JDBC driver	✘
Stored procedures	✘
Distributie en betrouwbaarheid	
Automatic horizontal partitioning	Ja door middel van Consistent Hashing. Gebaseerd op Amazon Dynamo (40) (45).
Automatic vertical partitioning	
Automatic Sharding	
Replication	✔ (45)
Referential integrity	✘
Transactions	✘

[5] "The Search, Network, and Analytics team at LinkedIn works on LinkedIn's information retrieval systems, the social graph system, data driven features, and supporting data infrastructure." - http://sna-projects.com/sna/

Integrity model	BASE
Database	
Ondersteuning datatypen	NUMBER, STRING, BOOLEAN, OBJECT, ARRAY (45).
Max. grootte db	Voor zover bekend, onbeperkt.
Max. grootte db tabel	Voor zover bekend, onbeperkt.
Max. grootte db rij	
Max. aantal kolommen per rij	
Opmerkingen	
• Relatief jong product. • Relatief weinig documentatie / kleine community.	
Meest geschikt voor	
• Verwerking van grote hoeveelheden data. • Toepassingen waarbij data integriteit geen vereiste is. • Gedistribueerde toepassingen. • Toepassingen waarbij ondersteuning van gemakkelijke, geavanceerde queries van ondergeschikt belang zijn ten opzichte van de performance.	

5.5.1 Conclusie

Project Voldemort is een *Available, Partition-Tolerant* DBMS en is gebaseerd op het K*ey-Value* datamodel. Voldemort is geschikt voor gedistribueerde omgevingen waarbij de consistentie van data van ondergeschikt belang is ten opzichte van de performance.

5.6 Oracle Database

Oracle Database is een commercieel databasemanagementsysteem. Ontwikkeld door Oracle en in gebruik bij LinkedIn. Binnen NCIM zijn meerdere professionals op het gebied van Oracle producten. Het gebruik door LinkedIn en de Oracle kennis binnen NCIM maakt dit DBMS een geschikte kandidaat voor het onderzoek.

Kenmerken

Algemeen	
Naam	Oracle Database
Ontwikkelaar	Oracle Corporation
Eerste release	1979
Laatste stable release	11g Release 2 / 11.2.0.2 (11-2010)
Release frequentie	Major releases ongeveer eens in de twee/drie jaar (47).
Geschreven in	C/C++
Programmeertaal compatibiliteit	OCI, OCCI, JDBC of ODBC ondersteunende talen zoals C, Java, PHP en .NET.
Besturingssysteem compatibiliteit	Windows, Linux, UNIX, Solaris, AIX en HP-UX.
Datamodel	Object-Relationeel (48)
Licentie	Commercieel
Website	http://www.oracle.com
Verbinding	
Embedded mode	✔ (49)
Client/server mode	✔
Querytaal	SQL
JDBC driver	✔
Stored procedures	✔ Ondersteuning voor Java Stored Procedures en PL/SQL (50).
Distributie en betrouwbaarheid	
Automatic horizontal partitioning	✔ (51)
Automatic vertical partitioning	
Automatic Sharding	
Replication	✔ *Multimaster Replication*, *Materialized View Replication* en *Hybrid replication* (52).
Referential integrity	✔ (53)
Transactions	✔ (54)
Integrity model	ACID (54)

Database	
Ondersteuning datatypen	Ondersteuning voor groot aantal numerieke, datum/tijd en tekst datatypen. Voor een compleet overzicht, zie: (55).
Max. grootte db	Onbeperkt (56)
Max. grootte db tabel	4 GB (56)
Max. grootte db rij	8 KB (56)
Max. aantal kolommen per rij	1000 kolommen (56)
Opmerkingen	
• Veel/duidelijke documentatie. • Hoge licentiekosten. • NCIM is Oracle Partner. • Binnen NCIM is kennis aanwezig betreft Oracle databases. • Ondersteuning voor groot aantal features. • Bewezen product in de markt.	
Meest geschikt voor	
• Object georiënteerde applicaties waarbij data integriteit (ACID) een vereiste is. • Applicaties waarbij schaalbaarheid van belang is.	

5.6.1 Conclusie

Oracle Database is een databasemanagementsysteem gebaseerd op het Object-relationele datamodel. Dit maakt het DBMS geschikt voor gebruik bij, in object georiënteerde talen ontwikkelde, applicaties. Oracle Database ondersteund vele features en de *partitioning* en *sharding* ondersteuning maakt dit product ook geschikt voor gedistribueerde toepassingen. Oracle kent echter relatief hoge licentiekosten wat het product minder aantrekkelijk maakt voor projecten met een relatief laag budget. Wanneer stabiliteit, integriteit en schaalbaarheid van belang zijn is Oracle Database een zeer geschikte keuze.

5.7 Neo4J

De door Twitter gebruikte *graph database* FlockDB was door het gebrek aan documentatie en de onvolwassenheid van het product ongeschikt om te onderzoeken. Als alternatief is gezocht naar een andere graph database om te onderzoeken. Gezien de vele beschikbare documentatie op de officiële website en de Google resultaten op de term 'Graph database' (drie van de eerste vijf resultaten gaan over Neo4j) is gekozen om Neo4j te onderzoeken.

Kenmerken

Algemeen	
Naam	Neo4j
Ontwikkelaar	Windh Technologies.
Eerste release	2002
Laatste stable release	Versie 1.2 (02-12-2010)
Release frequentie	Een aantal keer per jaar.
Geschreven in	Java
Programmeertaal compatibiliteit	Java, PHP, Python, Ruby, Scala en Groovy.
Besturingssysteem compatibiliteit	Windows, Linux, UNIX, Mac OS X en andere Java Virtual Machine (JVM) ondersteunende besturingssystemen.
Datamodel	Graph
Licentie	AGPLv3 / Commercieel
Website	http://neo4j.org
Verbinding	
Embedded mode	Ja.
Client/server mode	Ja.
Querytaal	API, Gremlin.
JDBC driver	✗
Stored procedures	✗
Distributie en betrouwbaarheid	
Automatic horizontal partitioning	
Automatic vertical partitioning	✗
Automatic Sharding	
Replication	✓ (57)
Referential integrity	✓ Nodes met relaties kunnen niet verwijderd worden.
Transactions	✓ (58)
Integrity model	ACID (58)

Database	
Ondersteuning datatypen	Boolean, boolean[], byte, byte[], short, short[], int, int[], long, long[], float, float[], double, double[], char, char[], java.lang.String, string[] (59).
Max. grootte db	Onbeperkt. De limieten van het aantal *nodes*, *relaties* en *properties* in de database zijn voor ieder een aantal van vier biljoen (60).
Max. grootte db tabel	Niet van toepassing.
Max. grootte db rij	Niet van toepassing.
Max. aantal kolommen per rij	Niet van toepassing.
Opmerkingen	
• Duidelijke documentatie. • Relatief kleine community ten opzichte van bekende relationele databases. • Een van de populairste databases in zijn soort afgaande van de aandacht die het krijgt op bijvoorbeeld blogs ten opzichte van andere *graph* databases.	
Meest geschikt voor	
• Opslag van *boomstructuur*-achtig gerelateerde data, zoals relaties tussen gebruikers van sociale netwerken.	

5.7.1 Conclusie

Neo4J is een databasemanagementsysteem, gebaseerd op het *Graph* datamodel. Neo4J heeft duidelijk documentatie maar een relatief kleine community ten opzichte van traditionele relationele DBMS-en. Het gebrek aan ondersteuning van *patitioning* en *sharding* maakt dit product minder geschikt voor schaalbare, gedistribueerde toepassingen. Maar gezien het feit dat er ondersteuning is voor vier biljoen *nodes* en *properties* is dit gebrek voor meeste projecten niet van belang.

5.8 MongoDB

MongoDB is een *document-oriented* databasemanagementsysteem. Van de beschikbare document-oriented DBMS-en is MongoDB één van de populairste (gebaseerd op Google Trends) en kent de officiële MongoDB website uitgebreide documentatie. Gezien het feit dat het onderzoek nog geen document-orient DBMS bevatte, de populariteit van het product en de uitgebreide beschikbare documentatie is MongoDB in het onderzoek opgenomen.

Kenmerken

Algemeen	
Naam	MongoDB
Ontwikkelaar	10gen
Eerste release	9 december 2009
Laatste stable release	Versie 1.6.5 (12-09-2010)
Release frequentie	Ongeveer twee keer per jaar.
Geschreven in	C++.
Programmeertaal compatibiliteit	C, C#, C++, Erlang, Haskell, Java, Javascript, Perl, PHP, Python, Ruby, Scala (61)
Besturingssysteem compatibiliteit	Windows, Mac OS X, Linux, Solaris
Datamodel	Document-oriented
Licentie	AGPL v3.0 / commercieel
Website	http://mongodb.org/
Verbinding	
Embedded mode	✗
Client/server mode	✓
Querytaal	API (JSON achtig) (62).
JDBC driver	Via externe driver: Mongo JDBC (63).
Stored procedures	Gedeeltelijk, via stored javascript functies (64).
Distributie en betrouwbaarheid	
Automatic horizontal partitioning	✓ MongoDB ondersteund horizontal partitioning en sharding (65).
Automatic vertical partitioning	
Automatic Sharding	
Replication	✓ Master-Slave Replication voor backup en verhoging van de beschikbaarheid (66).
Referential integrity	✗
Transactions	MongoDB biedt *Atomic* transactions voor een enkel document. De *single*

	transactions voldoen echter niet aan de overige ACID eisen (67).
Integrity model	Gedeeltelijk ACID (67).
Database	
Ondersteuning datatypen	Onder andere: String, integer, boolean, double, null, array, object, date, binary data (68).
Max. grootte db	24.000 *collections* (69), 2gb database limiet voor 32bit systemen (onbeperkt voor 64bit) (70).
Max. grootte db tabel	Niet van toepassing.
Max. grootte db rij	Niet van toepassing.
Max. aantal kolommen per rij	Niet van toepassing.
Opmerkingen	
• Veel/duidelijke documentatie. • Nieuw, populair product. Al veel gebruikt door grote partijen zoals Foursquare en SourceForge (71). • Geavanceerde query API ten opzichte van meeste andere NoSQL databases.	
Meest geschikt voor (72)	
• Als c*aching* toepassing. • Applicaties waarbij met grote hoeveelheden data wordt gewerkt. • *Real time analytics* / *logging*. • Schaalbare toepassingen waarbij de data altijd volledig consistent moet zijn.	

* Geschikt voor prototyping dankzij datamodel

5.8.1 Conclusie

MongoDB is een *Consistent, Partition-Tolerant* databasemanagementsysteem, gebaseerd op het *Document-oriented* datamodel. De standaard ondersteuning voor *horizontal partitioning* en *sharding* maakt MongoDB een geschikte keuze voor gedistribueerde toepassingen waarbij dataconsistentie een vereiste is. De uitgebreide query API geeft MongoDB een voordeel ten opzichte van andere NoSQL databases en maakt MongoDB een geschikte keus wanneer ingewikkelde queries uitgevoerd dienen te worden.

5.9 PostgreSQL

PostgreSQL is een veel gebruikt databasemanagementsysteem, volledig ontwikkeld door de community, opensource en wordt ondersteund door het bedrijfsleven.

Kenmerken

Algemeen	
Naam	PostgreSQL
Ontwikkelaar	Steunt op een wereldwijde gemeenschap van ontwikkelaars en bedrijven (73).
Eerste release	01-05-1995
Laatste stable release	Versie 9.0.3 (31-01-2011)
Release frequentie	Ongeveer één keer per jaar.
Geschreven in	C.
Programmeertaal compatibiliteit	C/C++, Java, .Net, Perl, Python, Ruby, Tcl en overige ODBC ondersteunende programmeertalen (74).
Besturingssysteem compatibiliteit	Voornamelijke gericht op FreeBSD, Linux, Mac OS X, Solaris en Windows (75).
Datamodel	Object-Relational
Licentie	PostgreSQL License (76).
Website	http://www.postgresql.org
Verbinding	
Embedded mode	✗
Client/server mode	✓
Querytaal	SQL
JDBC driver	✓ (77)
Stored procedures	✓ (74)
Distributie en betrouwbaarheid	
Automatic horizontal partitioning	Alleen ondersteuning voor horizontal partitioning door middel van *Range Partitioning* en *List Partitioning* (74).
Automatic vertical partitioning	
Automatic Sharding	
Replication	Ondersteuning voor vele type van replication zoals *Asynchronous Multimaster Replication* (78).
Referential integrity	✓ (78)
Transactions	✓ (74)
Integrity model	MVCC / ACID

Database	
Ondersteuning datatypen	Meeste SQL:2008 data typen, inclusief INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL en TIMESTAMP. Ook ondersteuning voor opslag van grote binary objecten zoals plaatjes, geluiden en video (74).
Max. grootte db	Onbeperkt (74).
Max. grootte db tabel	32 TB (74).
Max. grootte db rij	1,6 TB (74).
Max. aantal kolommen per rij	250 – 1600, afhankelijk van kolom type (74).
Opmerkingen	
• Veel/duidelijke documentatie. • Grote community. • Ondersteuning voor groot aantal features. • Bewezen product in de markt.	
Meest geschikt voor	
• Object georiënteerde toepassingen waarbij de integriteit van data van belang is. • Applicaties waarbij het wenselijk is om met opensource software te werken. • Opslag van grote hoeveelheden data.	

5.9.1 Conclusie

PostgreSQL is een databasemanagementsysteem, gebaseerd op het *Object-Relationele* datamodel. Dit maakt het DBMS geschikt voor gebruik bij, in object georiënteerde talen ontwikkelde, applicaties. De missende *sharding* ondersteuning maakt het DBMS minder geschikt voor gedistribueerde toepassen. De ondersteuning van *horizontal partitioning* en grote database grootten maakt PostgreSQL geschikt voor toepassingen waarbij gewerkt wordt met grote hoeveelheden data.

5.10 Microsoft SQL Server

Gezien het feit dat NCIM tot de Gold Partners van Microsoft behoort en binnen NCIM kennis aanwezig is omtrent Microsoft SQL Server is er voor gekozen dit DBMS op te nemen in het onderzoek. Voor dit onderzoek is gekozen voor de enterprise licentie gezien het feit dat lagere licenties geen ondersteuning bieden voor uitgebreide partitioning (79).

Kenmerken

Algemeen	
Naam	Microsoft SQL Server 2008 R2 Enterprise
Ontwikkelaar	Microsoft
Eerste release	1989
Laatste stable release	SQL Server 2008 R2 Enterprise (21-04-2010)
Release frequentie	Ongeveer eens in de twee jaar (80).
Geschreven in	C, C++, C#
Programmeertaal compatibiliteit	ASP, C, C++, COBOL, Perl, Python NET, Java, PHP, VBA, JavaScript (81).
Besturingssysteem compatibiliteit	Windows.
Datamodel	Relationeel
Licentie	Microsoft EULA
Website	www.microsoft.com/sqlserver
Verbinding	
Embedded mode	✔
Client/server mode	✔
Querytaal	SQL.
JDBC driver	✔ (81).
Stored procedures	✔ (82).
Distributie en betrouwbaarheid	
Automatic horizontal partitioning	Allen worden ondersteund door Microsoft SQL Server (83).
Automatic vertical partitioning	
Automatic Sharding	
Replication	✔ Voor backup en verhoging van de beschikbaarheid (84).
Referential integrity	✔ (83)
Transactions	✔ Transactions voldoen aan de ACID norm.
Integrity model	ACID
Database	

Ondersteuning datatypen	user-defined, sql_variant, xml, datetimeoffset, datetime2, datetime, smalldatetime, date, time, float, real, decimal, money, smallmoney, bigint, int, smallint, tinyint, bit, ntext, tekst, image, timestamp, uniqueidentifier, nvarchar, nchar, varchar, char, varbinary, binary (85).
Max. grootte db	524 PB (79).
Max. grootte db tabel	524 PB (18).
Max. grootte db rij	Onbeperkt (18).
Max. aantal kolommen per rij	30.000 kolommen (18).

Opmerkingen

- Veel documentatie.
- Documentatie onduidelijk verspreid over vele bronnen.
- Hoge licentiekosten.
- Bewezen product in de markt.
- Ondersteuning voor groot aantal features.
- Wanneer minder functionaliteiten benodigd zijn kan er gekozen worden om de gratis *express* editie te gebruken.
- Alleen Windows compatible.

Meest geschikt voor

- Toepassingen waarbij schaalbaarheid van belang is.
- Applicaties waarbij data integriteit (ACID) een vereiste is.
- Situaties waarbij een Windows server voorhanden is.

5.10.1 Conclusie

Microsoft SQL Server is een databasemanagementsysteem, gebaseerd op het relationele datamodel. De ondersteuning voor onder andere *partitioning* en *sharding* maakt MS SQL Server een geschikte keus wanneer de schaalbaarheid van data van belang is. Verder heeft MS SQL Server ondersteuning voor *ACID transactions* en wordt de integriteit van data door middel van *referential integrity* gegarandeerd. Gezien het feit dat het product zich inmiddels heeft bewezen als stabiel product in de markt en ondersteuning biedt voor een groot aantal features, kan geconcludeerd worden dat Microsoft SQL Server een goede keus is voor Business *Applications* waarbij schaalbaarheid en integriteit van data een vereiste is. Microsoft SQL Server kent echter licentiekosten en heeft slechts ondersteuning voor het Windows platform wat het product in sommige situaties ongeschikt maakt.

5.11 Apache CouchDB

Apache CouchDB is naast MongoDB één van de populairste *document-oriented* databasemanagementsystemen (gebaseerd op Google Trends). Gezien de uitgebreide beschikbare documentatie, de populariteit van het product en om MongoDB te kunnen vergelijken is CouchDB in het onderzoek opgenomen.

Kenmerken

Algemeen	
Naam	Apache CouchDB
Ontwikkelaar	Apache Software Foundation
Eerste release	2005. In 2008 werd CouchDB een Apache project.
Laatste stable release	Versie 1.0.2 (27-01-2011)
Release frequentie	Ongeveer vijf kleine en één grote release per jaar.
Geschreven in	Erlang
Programmeertaal compatibiliteit	Alle talen waar HTTP requests ondersteund worden. Er zijn, gemakkelijker te gebruiken, drivers beschikbaar voor onder andere: Java, C, Ruby, Python, PHP, Perl, Objective-C etc. (86).
Besturingssysteem compatibiliteit	Onder andere Android, Mac OS X, BSD Unix, Linux, Solaris en Windows (87).
Datamodel	Document-oriented
Licentie	Apache License 2.0
Website	http://couchdb.apache.org
Verbinding	
Embedded mode	✗
Client/server mode	✓
Querytaal	RESTful HTTP/JSON API (88).
JDBC driver	✗
Stored procedures	Gedeeltelijk, via *views* (89).
Distributie en betrouwbaarheid	
Automatic horizontal partitioning	Momenteel geen ondersteuning voor partitioning en sharding zonder gebruik van externe software. Er is een open source *fork* van CouchDB die het DBMS hebben uitgebreid met partitioning/sharding: Cloudant BigCouch (90).
Automatic vertical partitioning	
Automatic Sharding	
Replication	✓ Bi-directionele replication over meerdere servers met conflict handling (91).
Referential integrity	✗
Transactions	Gedeeltelijk: ACID *single document transactions*.

Integrity model	MVCC (92)
Database	
Ondersteuning datatypen	JSON types zoals: string, integer, boolean, date, array, map.
Max. grootte db	Onbekend (93).
Max. grootte db tabel	Niet van toepassing.
Max. grootte db rij	Niet van toepassing.
Max. aantal kolommen per rij	Niet van toepassing.
Opmerkingen	
• Veel/duidelijke documentatie. • In korte tijd, grote community (94). • Relatief nieuw product.	
Meest geschikt voor	
• Verwerking van grote hoeveelheden sparsed data. • Applicaties waarbij versiebeheer van belang is zoals bij *content management systems* en andere samenwerkings software zoals Wikipedia / SVN. • *Non-distributed* omgevingen.	

5.11.1 Conclusie

CouchDB is een *Available, Partition-Tolerant* databasemanagementsysteem gebaseerd op een *document-oriented* datamodel. De missende ondersteuning voor *partitioning* en *sharding* maakt het product ongeschikt voor gedistribueerde toepassingen, voor dat soort toepassingen kan beter naar *Cloudant BigCouch* uitgeweken worden. De ondersteuning voor *Multiversion concurrency control* (MVCC) maakt CouchDB een geschikte kandidaat voor toepassingen waarbij met meerdere versies van data gewerkt moet kunnen worden.

5.12 Redis

Redis is een populair Key-value databasemanagementsystemen in gebruik door een aantal grote partijen: als Craigslist, Digg en The Guardian (95). Door die reden is het DBMS in het onderzoek opgenomen.

Kenmerken

Algemeen	
Naam	Redis
Ontwikkelaar	Citrusbyte
Eerste release	Februari 2009
Laatste stable release	Versie 2.2.1 (23-02-2011)
Release frequentie	Ongeveer twee *major* releases per jaar.
Geschreven in	ANSI C
Programmeertaal compatibiliteit	Onder andere C, C++, PHP, Java, Erlang, Perl, Python, Ruby. Allen zijn API's die communiceren via het Redis protocol (96).
Besturingssysteem compatibiliteit	Onder andere Linux, BSD, OS X en Solaris. Geen officiële ondersteuning voor Windows (97).
Datamodel	Key-value
Licentie	BSD
Website	http://redis.io
Verbinding	
Embedded mode	✗
Client/server mode	✓
Querytaal	API
JDBC driver	✓ Via onofficiële jdbc-redis (98).
Stored procedures	✗
Distributie en betrouwbaarheid	
Automatic horizontal partitioning	Momenteel experimenteel en ongeschikt voor productie (99).
Automatic vertical partitioning	
Automatic Sharding	
Replication	✓ Master-Slave Replication voor backup en verhoging van de beschikbaarheid (100).
Referential integrity	✗
Transactions	✓ (101)
Integrity model	Volledig *Atomic*, *Consistent* en *Isolated* (van ACID norm). *Durability* is instelbaar: eens per x aantal sec. of voor elk schrijfcommando. Hoe hoger

	de *durability* eis, hoe lager de performance (102).
Database	
Ondersteuning datatypen	Binary-safe strings, List<binary-safe-string>, Set<binary-safe-string>, Sorted set (103).
Max. grootte db	150 miljoen *keys* per server (104).
Max. grootte db tabel	Niet van toepassing.
Max. grootte db rij	Niet van toepassing.
Max. aantal kolommen per rij	Niet van toepassing.
Opmerkingen	
• Redis is *single threaded*: om multi-core processors te benutten dienen er meerdere Redis instanties op de server gestart te worden. • Veel/duidelijke documentatie. • Ondersteund door VMware.	
Meest geschikt voor	
• Caching toepassingen, waarbij veel gebruikte data in geheugen wordt opgeslagen voor snelle *leesacties*. • Toepassingen waarbij de dataset kleiner is dan het beschikbare RAM geheugen op de server. • Toepassingen waarbij meer *read* dan *write* acties worden uitgevoerd. • Toepassingen waarbij de *durability* van data van ondergeschikt belang is ten opzichte van de performance.	

5.12.1 Conclusie

Redis is een *Consistent, Partition-Tolerant* databasemanagementsysteem gebaseerd op een *Key-value* datamodel. Gezien het feit dat alle data in RAM geheugen wordt opgeslagen – diskopslag is mogelijk maar traag – maakt het DBMS geschikt als *caching* toepassing voor veel opgevraagde data. Wanneer echter meer data opgeslagen moet worden dan dat RAM geheugen aanwezig is en alle data ongeveer even vaak wordt gelezen is Redis geen goede keuze.

5.13 AllegroGraph® RDFStore

AllegroGraph® RDFStore is een databasemanagementsysteem gebaseerd op een *RDF* en *Graph* model. Om Neo4J met een andere graph DBMS te kunnen vergelijken is RDFStore in het onderzoek opgenomen. Naast die reden is RDFStore één van de weinige stabiele graph DBMS-en en kent het uitgebreide documentatie.

Kenmerken

Algemeen	
Naam	AllegroGraph® RDFStore
Ontwikkelaar	Franz, Inc.
Eerste release	2004
Laatste stable release	Versie 4.2 (2011)
Release frequentie	Eens per jaar.
Geschreven in	Onbekend
Programmeertaal compatibiliteit	Java, Python, Sesame, Jena, Lisp, Ruby, Perl, C#, Clojure en Scala (105).
Besturingssysteem compatibiliteit	Linux x86-64 bit, versie 3.3 heeft bredere ondersteuning (106).
Datamodel	RDF / Graph
Licentie	Commercieel (107)
Website	http://www.franz.com/agraph/allegrograph
Verbinding	
Embedded mode	✗
Client/server mode	✓
Querytaal	SPARQL
JDBC driver	✗
Stored procedures	✓
Distributie en betrouwbaarheid	
Automatic horizontal partitioning	
Automatic vertical partitioning	✗
Automatic Sharding	
Replication	✓ Voor *load balancing* en schaalbaarheid.
Referential integrity	✓
Transactions	✓
Integrity model	ACID
Database	

Ondersteuning datatypen	numeric, dates, times, longitudes, latitudes, durations, telephone numbers, strings (106).
Max. grootte db	Onbekend
Max. grootte db tabel	Niet van toepassing.
Max. grootte db rij	Niet van toepassing.
Max. aantal kolommen per rij	Niet van toepassing.

Opmerkingen
• Uitgebreide documentatie. • Minder uitgebreide technische/architectuur documentatie. • Weinig externe informatie aanwezig.
Meest geschikt voor
• Opslag van *boomstructuur*-achtig gerelateerde data, zoals relaties tussen gebruikers van sociale netwerken. • Toepassingen met een RDF model.

5.13.1 Conclusie

AllegroGraph RDFStore is een databasemanagementsysteem met ondersteuning voor een *Graph* en *RDF* datamodel. Gezien de weinig beschikbare externe informatie en het *closed source* model van het product maakt AllegroGraph RDFStore een ongeschikt product voor gebruik in zakelijke toepassingen. Wanneer een *graph* datamodel wenselijk is, kan beter gekeken worden naar Neo4J.

5.14 Evaluatie

Zoals al bleek tijdens het eerdere onderzoek naar het DBMS gebruik bij sociale netwerken gebruiken Twitter, Facebook en LinkedIn alleen een combinatie van *relationele* en *column-oriented* databasemanagementsystemen. Het gebruik van de *column-oriented* DBMS-en HBase, Cassandra en Voldemort is gemakkelijk te verklaren: alle drie zijn zeer geschikt bij de verwerking van grote hoeveelheden data. De *column-oriented* oplossingen kennen echter geen uitgebreide API om complexe queries uit te kunnen voeren. Zowel MySQL als Oracle zijn inmiddels bewezen en stabiele producten op de markt en ondersteunen beide SQL als querytaal. MySQL en Oracle worden dus waarschijnlijk gebruikt daar waar complexe queries nodig zijn.

Aan de hand van de gestelde SNScanner criteria kunnen we de volgende selectie maken van DBMS-en, die interessant zijn om tijdens de performancetest op prestaties te gaan testen:

1. Schaalbaarheid: de *column-oriented* DBMS-en HBase, Cassandra en Voldemort, het relationele Oracle en het *document-oriented* MongoDB zijn zeer geschikt voor de verwerking van grote hoeveelheden data.

2. Snel lezen: moet aangetoond worden tijdens uit te voeren performancetest.

3. Veel relaties: Neo4j is hier als *Graph* oplossing zeer geschikt voor. Relationele DBMS-en bieden met behulp van *foreign keys* ook goede ondersteuning om relaties in de database te onderhouden. Bij relationele DBMS-en is voor het opslaan van relaties echter een zogenoemde koppeltabel benodigd.

4. Dynamische data en uitbreidbaarheid: MongoDB en CouchDB bieden met behulp van het *document-oriented* datamodel ondersteuning voor een dynamisch schema (dynamische data structuur).

5. Ondersteuning voor Java: alle onderzochte DBMS-en ondersteunen Java.

6. Gebruik van data van sociale netwerken: MySQL wordt door Twitter, Facebook en LinkedIn gebruikt en blijft dus interessant voor verder onderzoek.

Alle eigenschappen van de databasemanagementsystemen zijn achterhaald en er is een selectie gemaakt van eventueel geschikte DBMS-en. Voordat er echter een geschikte keuze voor het SNScanner project kan worden gemaakt, dienen de geselecteerde DBMS-en - met behulp van de benchmarktool - op performance getest te worden.

6 Performancetest van DBMS-en

Omdat onder andere de leessnelheid van het nieuwe databasemanagementsysteem een gestelde requirement was voor het SNScanner project, moest de performance van de verschillende onderzochte DBMS-en getest worden. Dit hoofdstuk beschrijft de testopstelling en de testresultaten van de uitgevoerde performancetest met behulp van de benchmarktool.

6.1 Doelstelling en aanpak

De doelstelling van de performancetest luidt:

- Achterhaal de performance van de onderzochte databasemanagementsystemen met behulp van de ontwikkelde benchmarktool.

6.1.1 Gelijkwaardige testomgeving

Bij het vergelijken en testen van zaken moet altijd in acht genomen worden dat het juiste, eerlijk wordt getest. Voor de performancetest betekent dit concreet: de databasemanagementsystemen moeten getest worden in een gelijkwaardige testomgeving, zodat externe factoren geen invloed hebben op de testresultaten.

Om voor elk databasemanagementsysteem een gelijke testomgeving te creëren zijn bij het testen de volgende maatregelen genomen:

1. Omdat elke testcasus – door het aanmaken van objecten - invloed heeft op het Java heap geheugen en omdat de Java *garbage collector* geen enkele garanties biedt tot het opruimen van de niet langer gebruikte objecten op de heap (108) wordt elke testcasus uitgevoerd in een nieuwe Java Virtual Machine (JVM). Het starten van een nieuwe JVM zorgt ervoor dat elke testcasus in een *frisse omgeving* wordt getest, zodat testresultaten elkaar niet beïnvloeden.

2. Elke testcasus is tien keer uitgevoerd, zodat externe factoren minder invloed hebben op de testresultaten. Met externe factoren worden hier de door Windows benodigde processen bedoeld die invloed kunnen hebben op de testresultaten. Niet essentiële processen – zoals virusscanner of firewall – zijn uitgeschakeld om invloed van externe factoren zoveel mogelijk uit te sluiten.

3. Voor het starten van de meetresultaten is een opwarmronde uitgevoerd. De opwarmronde zorgt ervoor dat het laden van de Java klassen door de Java Virtual Machine geen invloed heeft op de testresultaten (Meer over Java *Class Loading:* (109)).

6.1.2 Vergelijken van standaard instellingen

Op de databasemanagementsystemen kunnen veelal vele optimalisaties uitgevoerd worden. Configuratiesettings van het databasemanagementsysteem kunnen bijvoorbeeld geoptimaliseerd worden voor specifieke situaties. Omdat er echter geen tijd was om al deze optimalisaties uit te kunnen voeren en het gedeeltelijk optimaliseren van de DBMS-en tot oneerlijke resultaten zou leiden, is besloten om de databasemanagementsystemen tijdens de performancetest te gaan testen met alle standaard geleverde instellingen. Voor de performancetest betekende dit dat de resultaten aantoonden wat het snelste databasemanagementsysteem was wanneer gebruik werd gemaakt van

de standaard instellingen. Wanneer er achterhaald moet worden welk geoptimaliseerd DBMS het snelste is, moet de performancetest herhaald worden. Dit onderzoek viel echter buiten de scope van het opdracht maar kan door de flexibiliteit van de benchmarktool in de toekomst relatief gemakkelijk getest worden.

6.2 Testsysteem

Voor het testsysteem wordt er gebruik gemaakt van hardware, beschikbaar gesteld door de opdrachtgever. Het volgende testsysteem is ter beschikking gesteld voor het onderzoek:

Systeem	**Dell Latitude D830 X86 PC**
CPU	Intel™ Core™2 Duo CPU T7100 @ 1.80GHz, 1801 Mhz, 2 Core(s), 2 Logical Processor(s)
Total Physical Memory	2,00 GB
BIOS Version/Date	Dell Inc. A02, 7-6-2007
SMBIOS Version	2.4
Besturingssysteem	Microsoft Windows 7 Enterprise Version 6.1.7600 Build 7600

Om de performancetest op een later moment te kunnen reproduceren, dient er gebruik te worden gemaakt van een systeem met dezelfde eigenschappen. Er kan ook voor gekozen worden om alle tests op een ander systeem, bijvoorbeeld met een Linux besturingssysteem, te herhalen om zo de performance in samenwerking met Linux te kunnen testen.

Om elk DBMS in een gelijkwaardige testomgeving te testen, is elke test met dezelfde, minimaal benodigde Windows processen uitgevoerd.

6.3 Benchmarktool

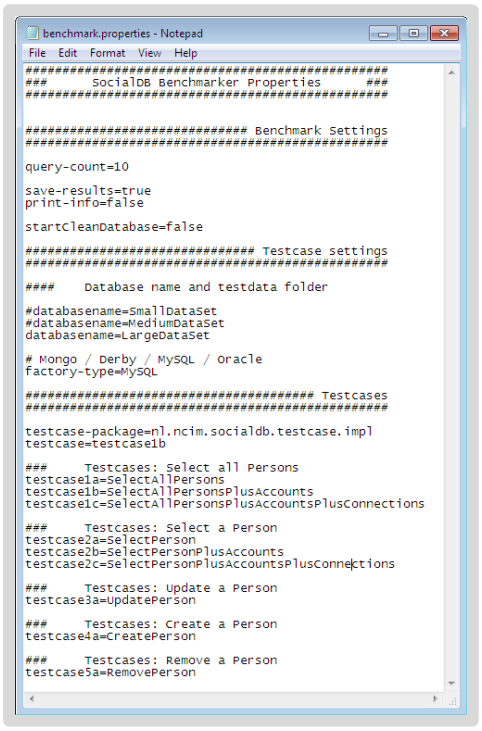
Dit hoofdstuk beschrijft de architectuur en de werking van de benchmarktool. Tot slot wordt in het hoofdstuk beschreven hoe een testcasus op een DBMS uitgevoerd kan worden.

6.3.1 Architectuur

Figuur 13 toont het klassendiagram van de benchmarktool. Onderstaande paragrafen beschrijven de belangrijkste individuele packages/klassen uit dit klassendiagram.

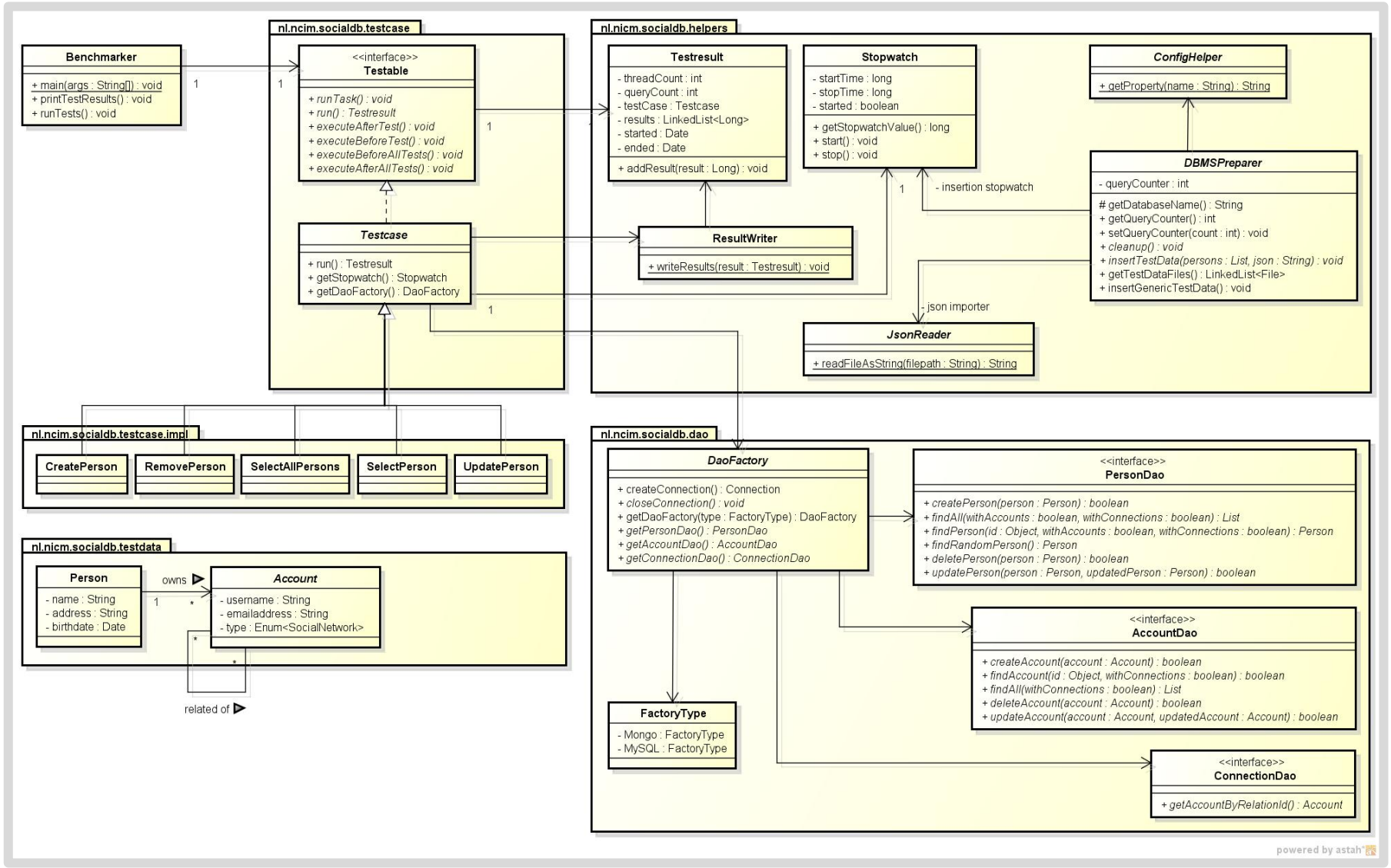
Properties bestand en ConfigHelper.java

Het properties bestand is een opsomming van eigenschappen die onder andere de uit te voeren testcasus bepalen bij het uitvoeren van de benchmarktool. De ConfigHelper klasse is een hulp-klasse met één doel: het ophalen van eigenschappen, zoals gedefinieerd in het properties bestand.



Figuur 12 - Properties bestand met benchmarksettings

1. **Query-count** (int): bepaald hoe vaak de testcasus uitgevoerd dient te worden. Testresultaten zijn betrouwbaarder wanneer er wordt gekeken naar het gemiddelde testresultaat van meerdere uitvoeringen.
2. **Save-results** (boolean): worden de resultaten na uitvoering opgeslagen? Bij het testen van de benchmarktool kan dit uitgeschakeld worden (*false*).
3. **StartCleanDatabase** (boolean): wordt bij het starten van de benchmarktool de database opgeruimd?
4. **Databasename** (string): wat is de naam van de database waarop de testcasus wordt uitgevoerd?
5. **Factory-type** (string): wat is de naam van het DBMS wat gebruikt dient te worden voor de testcasus.
6. **Testcase-package** (string): de package waarin de testcasussen te vinden zijn. Wanneer er op een later moment nieuwe testcasussen geïmplementeerd worden, zit men niet vast aan een bepaalde package (betere uitbreidbaarheid).
7. **Testcase** (string): de testcasus die uitgevoerd dient te worden. Deze waarde wijst naar testcasussen die daaronder worden gedefinieerd. Testcase1a verwijst bijvoorbeeld naar een klasse met de naam *SelectAllPersons.java*.



Figuur 13 – Klassendiagram benchmarktool

Omschrijving klassen en werking

Benchmarker: de Benchmarker klasse is het *entry-point* van de applicatie, dit wil zeggen dat deze klasse de main-methode bevat waar de applicatie bij het starten begint met uitvoeren.

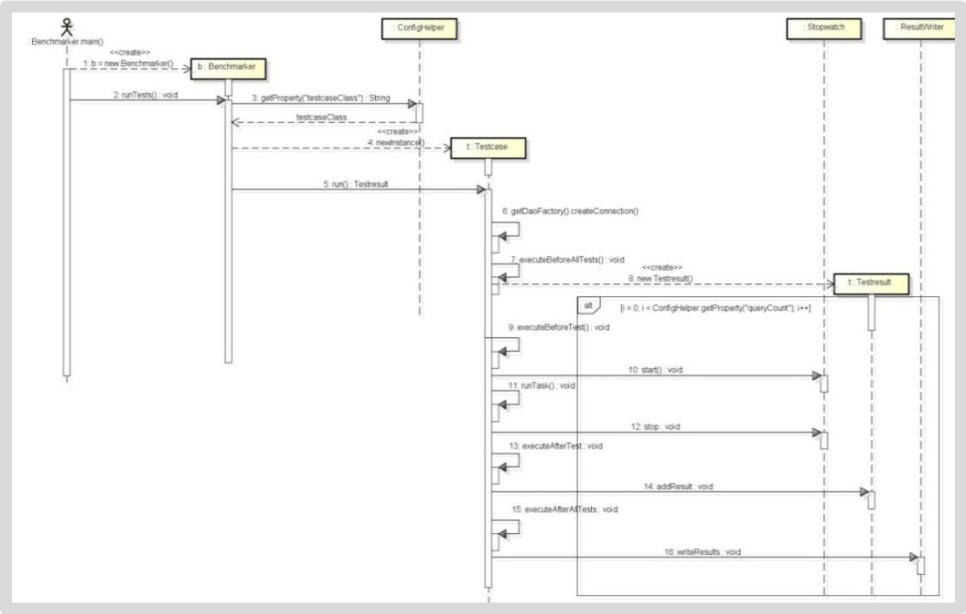
Testable en ***Testcase***: de package testcase bevat twee klassen: Testable (interface) en Testcase (abstract). Deze twee klassen vormen de basis voor de daadwerkelijke implementaties van de testcasussen en bepalen de basisstructuur waaraan een testcasus moet voldoen. In de main-methode van de Benchmarker klasse wordt een *Testcase* object gecreëerd waarvan vervolgens de run-methode wordt aangeroepen waarop de daadwerkelijke testcasus – zoals gespecificeerd in het properties bestand - wordt uitgevoerd.

DAO pattern *(Data Access Object)*: de package *.dao bevat de klasse *DaoFactory* die de verantwoordelijkheid heeft om verbinding te leggen met het juiste DBMS en aanroepen naar de juiste PersonDao, AccountDao en ConnectionDao implementaties te delegeren. Het achterhalen van de Dao implementaties en welke verbinding gecreëerd moet worden, gebeurt op basis van gedefinieerde waarde (*FactoryType*) in het properties bestand.

Helpers: de *.helpers package bestaat uit klassen die een ondersteunende rol hebben:

- ***Stopwatch***: meet de tijd tijdens het uitvoeren van een testcasus.
- ***Testresult***: het testresultaat wordt opgeslagen in een *Testresult* object.
- ***Resultwriter***: schrijft de testresultaten naar een bestand op de harde schijf van het testsysteem
- ***JsonReader***: leest de .json testdata bestanden uit.
- ***DBMSPreparer***: brengt het te testen DBMS in de originele staat zodat elke testcasus op dezelfde dataset wordt uitgevoerd.

Onderstaande sequentiediagram toont de werking van de benchmarktool tijdens de uitvoering van een testcasus:



Figuur 14 – Sequentiediagram werking benchmarktool

6.3.2 Handleiding: uitvoeren testcasus

Onderstaande paragrafen beschrijven stap voor stap hoe een testcasus uitgevoerd kan worden. Als voorbeeld gebruiken we *c:/databases* als standaard directory voor alle bestanden.

Installeer DBMS

Installeer het gewenste DBMS in de directory. Raadpleeg de bij het DBMS bijgeleverde documentatie voor hulp bij deze stap.

Bereid testomgeving voor

Download alle bestanden uit het NCIM SocialDBMS svn project en plaats de bestanden in de directory:

1. Plaats de jar-file *SocialDBMSBenchmarkV3.jar* in de directory.
2. Plaats de *testdata* folder in directory/testdata.
3. Bewerk het *benchmark.properties* bestand naar gewenste testinstellingen .

Start DBMS

Start het gewenste – zoals in het properties bestand is gedefinieerd – DBMS. Raadpleeg de bij het DBMS bijgeleverde documentatie voor hulp bij deze stap.

Start DBMSPreparer

Start de meegeleverde DBMS voorbereider - om de data in het DBMS te importeren - op de volgende wijze:

```
java -jar -Xms512m "c:/databases/*DBMS-naam*Preparer.jar"
```

Tijdens het voorbereiden van het DBMS wordt u van de voortgang op de hoogte gehouden. Wanneer voltooid, ontvangt u hier bericht van. Zie sectie 6.4 voor meer informatie over het genereren van de testdata.

Start benchmarktool

Start de benchmarktool door middel van de *command line*:

```
java -jar -Xms512m "c:/databases/SocialDBMSBenchmarkV3.jar"
```

Tijdens het uitvoeren van de testcasus wordt u van de voortgang op de hoogte gehouden. Wanneer voltooid, ontvangt u hier bericht van.

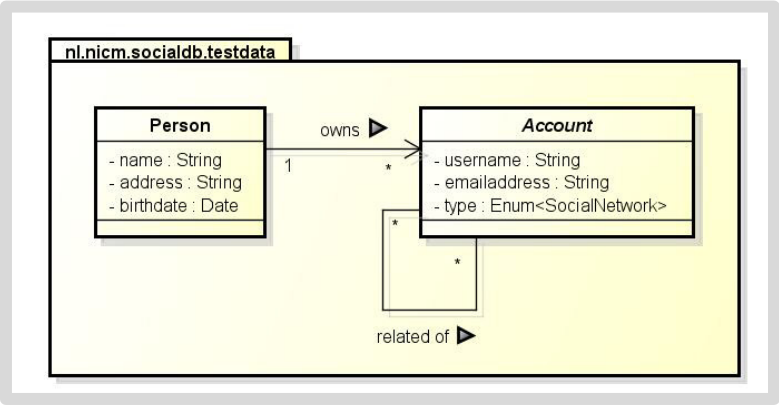
Testresultaten raadplegen

Wanneer er voor het opslaan van de testresultaten is gekozen – in het properties bestand – kunt u de resultaten van de uitgevoerde testcasus raadplegen in de folder: *c:/databases/results*.

6.4 Omschrijving testdata

6.4.1 Klassendiagram van de testdata

Omdat er een databasemanagementsysteem gekozen moet worden voor het SNScanner project komt de testdata, die tijdens het testen gebruikt gaat worden, overeen met die van het SNScanner project. Het klassendiagram van de gemodelleerde testdata is te zien in Figuur 15.



Figuur 15 - Klassendiagram testdata

De *Person*-klasse in het klassendiagram representeert een persoon met bijbehorende persoonsgegevens, zoals zijn geboortedatum en adres. Een *Person* kan nul of meerdere *Accounts* hebben van verschillende (*type*) sociale netwerken en elk account heeft weer connecties met andere accounts.

6.4.2 Verschillende grootte datasets

Omdat de resultaten van het onderzoek een breed draagvlak moeten hebben, de resultaten moeten ook ter ondersteuning gebruikt kunnen worden bij de besluitvorming voor DBMS keuze van toekomstige projecten, is er gekozen om de testcasussen bij drie verschillende grootte van datasets te testen. Door meerdere grootte van datasets te testen, zijn de resultaten geschikt ter naslag bij toekomstige projecten waarbij met kleine, middelgrote en grote hoeveelheden data gewerkt gaat worden.

1. De ***kleine*** dataset bestaat uit 100 personen met elk twee accounts, waarvan elk account tien connecties bevat naar andere accounts. De inhoud van de database ziet er dan als volgt uit:
    a. 100 personen
    b. 200 accounts
    c. 2.000 connecties

2. De ***middelgrote*** dataset bestaat uit 10.000 personen. Het aantal accounts per persoon en het aantal connecties per account blijven, ten opzichte van de kleine dataset, onveranderd. De inhoud van de database ziet er als volgt uit:
    a. 10.000 personen
    b. 20.000 accounts
    c. 200.000 connecties

3. De ***grote*** dataset bestaat uit 100.000 personen. Het aantal accounts per persoon en het aantal connecties per account blijven, ten opzichte van de kleine en middelgrote dataset, onveranderd. De inhoud van de database ziet er als volgt uit:
    a. 100.000 personen
    b. 200.000 accounts
    c. 2.000.000 connecties

6.4.3 Genereren van consistente testdata

Om consistente data te genereren die voor elke testcasus en alle DBMS-en te gebruiken zijn, is er een testdata generator ontwikkeld in Java. De testdata generator is onafhankelijk van de benchmarktool en is in staat om 'sociale' testdata te genereren volgens het testdata klassendiagram in Figuur 15. Elke dataset (klein, middelgroot en groot) wordt eenmalig gegenereerd zodat voor elk DBMS exact dezelfde testdata wordt gebruikt.

Formaat

De testdata generator produceert een vooringesteld aantal personen, accounts en connecties en slaat deze vervolgens in JSON formaat op als bestand. Er is voor het JSON formaat gekozen, omdat het zeer leesbaar is en gemakkelijk omgezet kan worden naar andere formaten.

> *JSON (JavaScript Object Notation)* (110)*: is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate.*

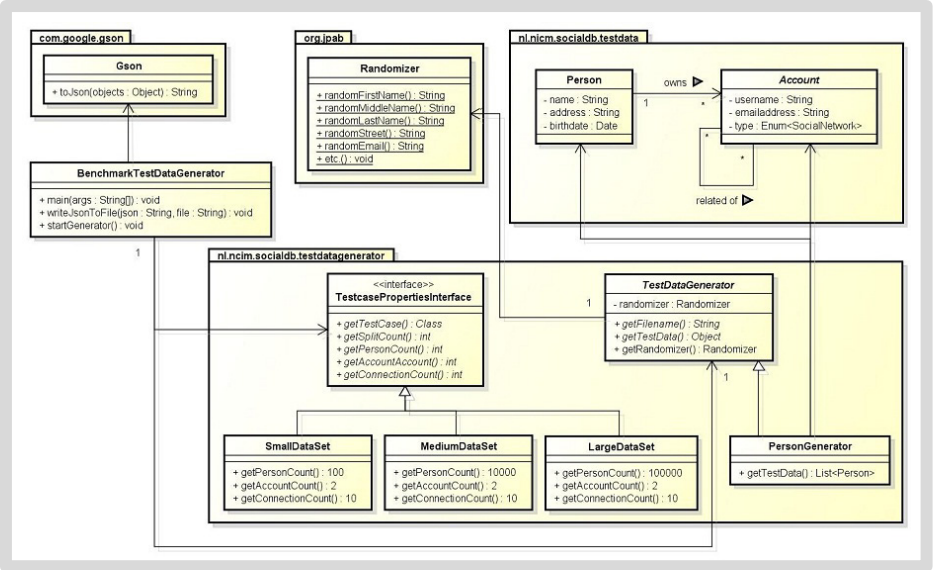
Een enkel persoon, met twee accounts en meerdere connecties ziet er in JSON formaat als volgt uit:

```
{
    "name": "Nick Groenewegen",
    "address": "ANVKCVEAGCD 18 191658052",
    "birthdate": "06-aug-1988",
    "accounts": [
      {
        "relationid": 199,
        "username": "IKDPRS",
        "emailaddress": "TWN@SHWDPX.XP",
        "type": "TWITTER",
        "connections": [2, 19,  Etc].
       ]
      },
      {
        "relationid": 200,
        "username": "RWDRXWFW",
        "emailaddress": "AAY@JFO.ONV",
        "type": "HYVES",
        "connections": "connections": [44, 55,  Etc].
      }
    ]
}
```

Figuur 16 – Testdata: enkel persoon in JSON formaat

Architectuur

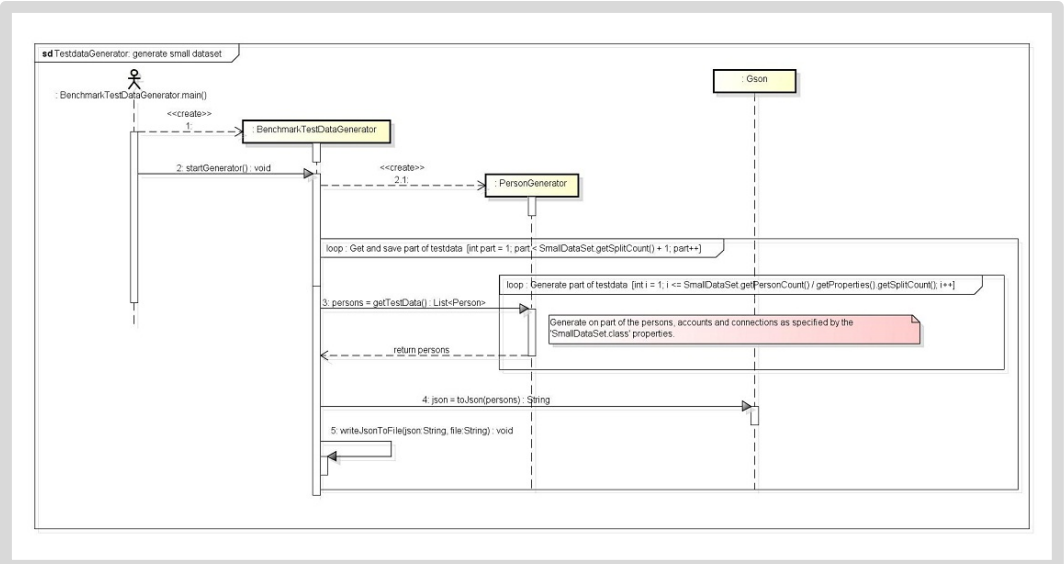
Het verkleinde klassendiagram in Figuur 17 toont de architectuur van de testdata generator. De package *com.google.gson* komt uit de externe *Gson* library en bevat klassen die objecten, in dit geval Person en Account objecten kan converteren naar JSON formaat. Ook bevat de Gson library functies om JSON naar Person en Account objecten om te zetten.



Figuur 17 – Klassendiagram testdata generator.

Werking

Het sequentiediagram in Figuur 18 beschrijft de werking van de testdata generator. Het diagram toont globaal hoe een kleine testdataset wordt gegenereerd. De werking van de testgenerator met betrekking tot het genereren van de middelgrote en grote dataset is hetzelfde. Het resultaat van het uitvoeren van de testdata generator is een bestand met als inhoud het vooringesteld aantal personen in JSON formaat.



Figuur 18 – Sequentiediagram testdata generator

6.5 Selectie van de databasemanagementsystemen

Tijdens de performancetest zijn vijf databasemanagementsystemen getest: Derby, MySQL, Oracle, MongoDB en CouchDB. Deze sectie beschrijft hoe tot deze selectie is gekomen.

De volgende redenen en de evaluatie in sectie 5.14 hebben tot de selectie van de eerste drie relationele DBMS-en geleid:

1. **Derby**: is het gekozen databasemanagementsysteem voor het SNScanner project. Tijdens de performancetest is bekeken of de keuze voor Derby (qua performance) de juiste was.
2. **MySQL**: is door alle onderzochte sociale netwerken in gebruik (zie hoofdstuk 4) en dus een logische keuze om op te nemen in de performancetest
3. **Oracle**: is in gebruik door LinkedIn en NCIM heeft professionals, gespecialiseerd in Oracle producten wat het DBMS interessant maakte om op te nemen in de performancetest.

Vervolgens stond, door het gebruik bij de sociale netwerken, de implementatie van *Cassandra, Voldemort en HBase* op de planning. Helaas waren voor al deze *column-oriented* DBMS-en geen goede Java connectoren beschikbaar. Communicatie vanuit een Java applicatie was alleen mogelijk via het http-protocol. Het is niet gelukt om de testcasussen met behulp van het http-protocol in de beschikbare tijd te implementeren. Omdat deze DBMS-en bij zowel Twitter, Facebook als LinkedIn gebruikt worden, is het nuttig om deze DBMS-en in de toekomst alsnog op performance te gaan testen.

Door de gestelde eis “dynamische datastructuur” (zie sectie 2.1.2 voor alle gestelde criteria) is gekozen om MongoDB en CouchDB te gaan implementeren. MongoDB en CouchDB zijn beide *document-oriented,* waardoor de structuur (dataschema) van de gegevens dynamisch is. Na afronding van de MongoDB en CouchDB implementatie is besloten om geen andere DBMS-en meer toe te voegen, zodat er genoeg tijd zou zijn om de al geïmplementeerde database-managementsystemen te testen.

6.6 Omschrijving testcasussen

Onderstaande testcasussen zijn uitgevoerd op de geteste databasemanagementsystemen. Voor het SNScanner project is vooral de leessnelheid van het DBMS belangrijk. Om deze reden zijn er zes testcasussen opgesteld die allen de leessnelheid testen. Omdat de schrijfsnelheid natuurlijk niet helemaal onbelangrijk is, zijn er drie voorkomende testcasussen opgesteld die de schrijfsnelheid van het DBMS toetsen: het verwijderen van data, het wijzigen van data en het toevoegen van data in de database.

Bij elke testcasus zijn – wanneer gebruikt - de volgende eigenschappen beschreven:

> *ExecuteBeforeAllTests:* deze zaken worden eenmalig voor het starten van elke testcasus uitgevoerd.
> *ExecuteBeforeTest*: deze zaken worden voor het uitvoeren van een enkele testrun uitgevoerd. Wanneer een testcasus tien keer wordt uitgevoerd, worden deze zaken ook tien keer uitgevoerd voor het starten van de testrun.
> *ExecuteAfterAllTests:* deze zaken worden eenmalig na het eindigen van elke testcasus uitgevoerd.

ExecuteAfterTest: deze zaken worden na het eindigen van een enkele testrun uitgevoerd. Wanneer een testcasus tien keer wordt uitgevoerd, worden deze zaken ook tien keer uitgevoerd na het eindigen van de testrun.

Bovenstaande activiteiten beschrijven wat er vóór de testcasus en na de testcasus is uitgevoerd om het databasemanagementsysteem voor te bereiden op de daadwerkelijke test. Omdat het voorbereidende zaken betreft, is de duur van deze activiteiten niet bij het testresultaat opgenomen. Bij elke testcasus is de corresponderende SQL query gegeven. De testcasussen van MongoDB en CouchDB zijn met behulp van de meegeleverde Java API's geïmplementeerd.

Sectie 6.7 beschrijft hoe de testcasussen met behulp van de benchmarktool zijn getest.

6.6.1 Alle personen ophalen

Deze testcasus haalt alle personen op uit de database. Deze testcasus is uitgevoerd om te achterhalen hoelang het duurt om een overzichtspagina van alle personen te genereren. Dit scenario komt niet veel voor tijdens het dagelijks gebruik van de SNScanner applicatie en zal dus minder meewegen bij de uiteindelijke DBMS keuze.

Haal alle personen op uit de database.	
Testcasus code	1a
Resultaat testcasus	Set van *Person* objecten, zonder *Accounts* en *zonder Relaties*
ExecuteAfterTest	Verifieer resultaat

Onderstaand codevoorbeeld toont de corresponderende SQL query van de testcasus.

```
SELECT *
FROM person AS person
ORDER BY person.id
```

Codevoorbeeld 1 - Testcasus 1a SQL implementatie

6.6.2 Alle personen met accounts ophalen

Deze testcasus haalt alle personen met bijbehorende accounts op uit de database. Deze testcasus is uitgevoerd om te achterhalen hoelang het duurt om een overzichtspagina van alle personen met accounts te genereren. Dit scenario komt niet veel voor tijdens het dagelijks gebruik van de SNScanner applicatie en zal dus minder meewegen bij de uiteindelijke DBMS keuze.

Haal alle personen met bijbehorende accounts op uit de database.	
Testcasus code	1b
Resultaat testcasus	Set van *Person* objecten, met *Accounts* en *zonder Relaties*
ExecuteAfterTest	Verifieer resultaat

Onderstaand codevoorbeeld toont de corresponderende SQL query van de testcasus.

```
SELECT
        person.*,
        account.*,
        account.id AS relationid

FROM person AS person

LEFT JOIN account AS account
        ON (person.id = account.person_id)
ORDER BY person.id
```

Codevoorbeeld 2 - Testcasus 1b SQL implementatie

6.6.3 Alle personen met accounts en relaties ophalen

Deze testcasus haalt alle personen met bijbehorende accounts en connecties op uit de database. Deze testcasus is uitgevoerd om te achterhalen hoelang het duurt om een overzichtspagina van alle personen met onderlinge connecties te genereren. Dit scenario komt regelmatig voor tijdens het dagelijks gebruik van de SNScanner applicatie en zal dus gemiddeld meewegen bij de uiteindelijke DBMS keuze.

Haal alle personen met bijbehorende accounts en relaties op uit de database.	
Testcasus code	1c
Resultaat testcasus	Set van *Person* objecten, met *Accounts* en *Relaties*
ExecuteAfterTest	Verifieer resultaat

Onderstaand codevoorbeeld toont de corresponderende SQL query van de testcasus.

```
SELECT
        person.*,
        account.*,
        account.id AS relationid,
        sconnection.*

FROM person AS person

LEFT JOIN account AS account
        ON (person.id = account.person_id)

LEFT JOIN connection AS sconnection
        ON (account.id = sconnection.account_id)

ORDER BY person.id
```

Codevoorbeeld 3 - Testcasus 1c SQL implementatie

6.6.4 Enkel persoon ophalen

Deze testcasus haalt een enkel persoon op uit de database. Deze testcasus is uitgevoerd om te achterhalen hoelang het duurt wanneer de persoonsgegevens van een enkel persoon opgevraagd dienen te worden. Dit scenario komt niet veel voor tijdens het dagelijks gebruik van de SNScanner applicatie en zal dus minder meewegen bij de uiteindelijke DBMS keuze.

Haal enkel persoon op uit de database op basis van een sleutel (zoals *primary key* bij relationele database).	
Testcasus code	2a
ExecuteBeforeAllTests	Bepaal de sleutel van een persoon op rij 50, zodat voor elk DBMS dezelfde persoon (zelfde eigenschappen) wordt gebruikt.
Resultaat testcasus	Enkel *Person* object, zonder *Accounts* en *zonder Relaties*
ExecuteAfterTest	Verifieer resultaat

Onderstaand codevoorbeeld toont de corresponderende SQL query van de testcasus.

```
SELECT *
FROM person AS person
WHERE person.id = ?
ORDER BY person.id
```

Codevoorbeeld 4 - Testcasus 2a SQL implementatie

6.6.5 Enkel persoon met accounts ophalen

Deze testcasus haalt een enkel persoon met bijbehorende accounts op uit de database. Deze testcasus is uitgevoerd om te achterhalen hoelang het duurt wanneer de persoonsgegevens en accountgegevens van een enkel persoon opgevraagd dienen te worden. Dit scenario komt niet veel voor tijdens het dagelijks gebruik van de SNScanner applicatie en zal dus minder meewegen bij de uiteindelijke DBMS keuze.

Haal enkel persoon met bijbehorende accounts op uit de database, op basis van een sleutel.	
Testcasus code	2b
ExecuteBeforeAllTests	Bepaal de sleutel van een persoon op rij 50, zodat voor elk DBMS dezelfde persoon (zelfde eigenschappen) wordt gebruikt.
Resultaat testcasus	Enkel *Person* object, met *Accounts* en *zonder Relaties*
ExecuteAfterTest	Verifieer resultaat

Onderstaand codevoorbeeld toont de corresponderende SQL query van de testcasus.

```
SELECT
        person.*,
        account.*,
        account.id AS relationid

FROM person AS person

LEFT JOIN account AS account
        ON (person.id = account.person_id)

WHERE person.id = ?

ORDER BY person.id
```

Codevoorbeeld 5 - Testcasus 2b SQL implementatie

6.6.6 Enkel persoon met accounts en relaties ophalen

Deze testcasus haalt een enkel persoon met bijbehorende accounts en relaties op uit de database. Deze testcasus is uitgevoerd om te achterhalen hoelang het duurt wanneer de persoonsgegevens, accountgegevens en connecties met andere personen van een enkel persoon opgevraagd dienen te worden. Dit scenario komt veel voor tijdens het dagelijks gebruik van de SNScanner applicatie en zal dus zwaar meewegen bij de uiteindelijke DBMS keuze.

Haal enkel persoon met bijbehorende accounts en relaties op uit de database op basis van een sleutel.	
Testcasus code	2c
ExecuteBeforeAllTests	Bepaal de sleutel van een persoon op rij 50, zodat voor elk DBMS dezelfde persoon (zelfde eigenschappen) wordt gebruikt.
Resultaat testcasus	Enkel *Person* object, met *Accounts* en *Relaties*
ExecuteAfterTest	Verifieer resultaat

Onderstaand codevoorbeeld toont de corresponderende SQL query van de testcasus.

```
SELECT
        person.*, account.*, account.id AS
        relationid, sconnection.*
FROM person AS person
LEFT JOIN account AS account
        ON (person.id = account.person_id)
LEFT JOIN connection AS sconnection
        ON (account.id = sconnection.account_id)

WHERE person.id = ?
ORDER BY person.id
```

Codevoorbeeld 6 - Testcasus 2c SQL implementatie

6.6.7 Verwijder persoon met accounts en relaties

Deze testcasus verwijdert een enkel persoon met bijbehorende accounts en relaties. Dit scenario komt niet veel voor tijdens het dagelijks gebruik van de SNScanner applicatie en zal dus minder meewegen bij de uiteindelijke DBMS keuze.

Verwijder enkel persoon met bijbehorende accounts en connecties op basis van een sleutel.	
Testcasus code	3
ExecuteBeforeTest	Bepaal de sleutel van een persoon op rij 50, zodat voor elk DBMS dezelfde persoon (zelfde eigenschappen) wordt gebruikt.
Resultaat testcasus	Succesvol verwijderd? Boolean waarde (true / false).
ExecuteAfterTest	Verifieer resultaat

Onderstaand codevoorbeeld toont de corresponderende SQL query van de testcasus.

```
DELETE FROM person WHERE id = ?
```

Codevoorbeeld 7 - Testcasus 3 SQL implementatie

Doordat in alle geteste DBMS-en foreign keys en constraints aangelegd zijn bij de relationele DBMS-en, worden bijbehorende accounts en connecties automatisch verwijderd na het verwijderen van een persoon.

6.6.8 Update persoon

Deze testcasus update de persoonsgegevens van een enkel persoon. Dit scenario komt niet veel voor tijdens het dagelijks gebruik van de SNScanner applicatie en zal dus minder meewegen bij de uiteindelijke DBMS keuze.

Update de persoonsgegevens van een persoon.	
Testcasus code	4
ExecuteBeforeAllTests	Bepaal de sleutel van een persoon op rij 50, zodat voor elk DBMS dezelfde persoon (zelfde eigenschappen) wordt gebruikt.
Resultaat testcasus	Succesvol geüpdatet? Boolean waarde (true / false).
ExecuteAfterTest	Verifieer resultaat

Onderstaand codevoorbeeld toont de corresponderende SQL query van de testcasus.

```
UPDATE person SET name = ? WHERE name = ?
```

Codevoorbeeld 8 - Testcasus 4 SQL implementatie

6.6.9 Voeg persoon toe

Deze testcasus slaat een nieuw persoon op in de database. Dit scenario komt niet veel voor tijdens het dagelijks gebruik van de SNScanner applicatie en zal dus minder meewegen bij de uiteindelijke DBMS keuze.

Voeg een persoon toe met bijbehorende accounts en relaties in de database.	
Testcasus code	5
ExecuteBeforeAllTests	Bepaal de sleutel van een persoon op rij 50, zodat voor elk DBMS dezelfde persoon (zelfde eigenschappen) wordt gebruikt.
ExecuteBeforeTest	Verander de *name*-waarde waarin de naam veranderd moet worden. Bij elke testrun wordt de testrun teller toegevoegd aan de *name*-waarde, zodat voor elke testrun geverifieerd kan worden of de update succesvol is.
Resultaat testcasus	Succesvol geüpdatet? Boolean waarde (true / false).
ExecuteAfterTest	Verifieer resultaat

Onderstaand codevoorbeeld toont de corresponderende SQL query van de testcasus.

```
INSERT INTO person  (name, address, birthdate)
    VALUES (?, ?, ?)
INSERT INTO account
    (id, person_id, username, emailaddress, type)
    VALUES (?, ?, ?, ?, ?)
INSERT INTO connection
    (account_id, account2_id) VALUES (?, ?)
```

Codevoorbeeld 9 - Testcasus 5 SQL implementatie

6.7 Handleiding benchmarktool: uitvoeren van een testcasus

Deze sectie beschrijft stap voor stap hoe een testcasus uitgevoerd kan worden met de benchmarktool. Als voorbeeld gebruiken we *c:/databases* als gekozen directory voor de bestanden. Deze handleiding is bedoeld om aan te tonen hoe de testcasussen zijn uitgevoerd met behulp van de benchmarktool.

6.7.1 Installeer DBMS

Installeer het DBMS wat getest dient te worden. Raadpleeg de bij het DBMS bijgeleverde documentatie voor hulp bij deze stap.

6.7.2 Bereid testomgeving voor

Download alle bestanden uit het NCIM SocialDBMS SVN-project (versiebeheer) en plaats alle bestanden in de gekozen directory. Onder de bestanden bevinden zich onder andere:

4. Het *Benchmarktool.jar* bestand wat het gecompileerde project bevat en via de command line gemakkelijk gestart kan worden;
5. De testdata bestanden;
6. En het *benchmark.properties* bestand.

Voor het starten van de benchmarktool moeten de juiste instellingen in het bestand aangepast worden, waaronder de belangrijkste drie: het te gebruiken DBMS (*Factory-type*), de te gebruiken dataset (*Databasename*) en de uit te voeren testcasus (*Testcase*).

6.7.3 Start DBMS

Start het gewenste – welke in het properties bestand is gedefinieerd – DBMS en importeer de testdata. Raadpleeg de bij het DBMS bijgeleverde documentatie voor hulp bij deze stap.

6.7.4 Start benchmarktool

Start de benchmarktool door middel van de *command line*:

```
java -jar "c:/databases/Benchmarktool.jar"
```

Tijdens het uitvoeren van de testcasus wordt de voortgang van de testcasus getoond. Wanneer de testcasus is voltooid, worden de testresultaten met bijbehorende gebruikte instellingen opgeslagen in de opgegeven directory.

6.7.5 Testresultaten raadplegen

De testresultaten kunnen geraadpleegd en met eerdere testresultaten vergeleken worden.

6.8 Testresultaten van de performancetest

Deze sectie beschrijft de gemeten testresultaten van alle uitgevoerde testcasussen op Oracle, MySQL, Derby, MongoDB en CouchDB. Bij elke testcasus zijn de testresultaten toegelicht aan de hand van een grafiek. De grafiek bevat voor alle DBMS-en drie testresultaten: de gemeten resultaten met de kleine, de middelgrote en de grote dataset (zie sectie 6.4.2). Zoals eerder vermeld is elke testcasus voor elke dataset tien keer uitgevoerd. De grafiek toont per dataset het gemiddelde testresultaat.

Op de x-as van de grafiek zijn de DBMS-en verdeeld over de drie grootte van datasets, op de y-as de duur van de query per persoon[6] (in milliseconde). Voor de overzichtelijkheid en leesbaarheid van de resultaten in de grafiek is gekozen voor een logaritmische y-as en zijn de as-titels (behalve voor eerste testcasus) weggelaten.

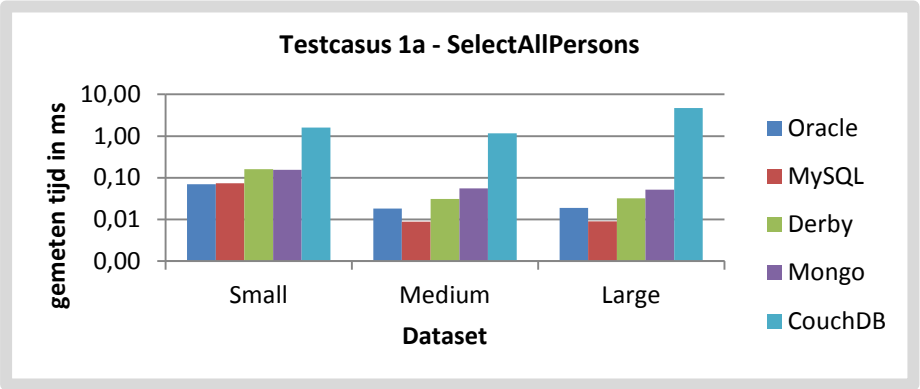
Bij elke testcasus is voor elk databasemanagementsysteem *nul* tot en met vier sterren toegekend. Een DBMS krijgt vier sterren toegekend bij een heel goed testresultaat en nul sterren bij een heel slecht resultaat. Zie de algehele conclusie en evaluatie van de performancetest voor het overzicht van de toegekende sterren (sectie 6.8.6).

6.8.1 Testcasus 1 – Alle personen ophalen

Deze sectie beschrijft de testresultaten van het ophalen van alle personen uit de database. De test is opgedeeld in de drie volgende concrete testcasussen:

1a: Het ophalen van alle personen.
1b: Het ophalen van alle personen met bijbehorende accounts.
1c: Het ophalen van alle personen met bijbehorende accounts en bijbehorende connecties.

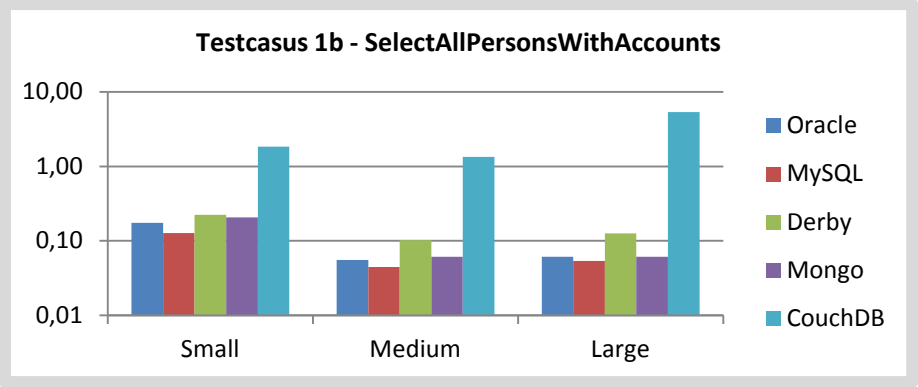
Onderstaande grafieken geven de testresultaten weer voor deze drie tests.



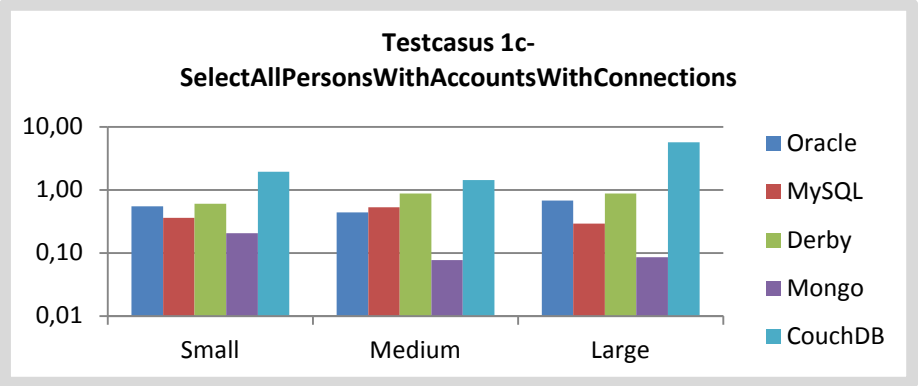
MySQL presteert het best over de drie datasets. Oracle komt als tweede uit de test, gevolgd door Derby, MongoDB en tot slot CouchDB. Bij alle vijf de DBMS-en is een verbetering te constateren van de 'Medium dataset'-meting ten opzichte van de Small dataset. De meetresultaten bij de Large dataset blijven bij MySQL, Oracle en Derby ten opzichte van de Medium dataset gelijk. MongoDB

[6] Wanneer de query alle personen van de dataset ophaalt, is het testresultaat *niet* de totale tijd maar de gemeten tijd *per* persoon. Bijvoorbeeld: een tijdsmeting van 10 seconde voor 1000 personen: resultaat = 10ms.

presteert nog beter bij de grote dataset. Daarentegen presteert CouchDB een factor vier slechter bij de Large dataset ten opzichte van de Medium dataset.



Ten opzichte van testcasus 1a worden nu naast alle personen ook alle bijbehorende accountgegevens opgehaald. Waar de resultaten van Oracle, MySQL en Derby ten opzichte van elkaar relatief gezien gelijk blijven, zijn de resultaten in verhouding met vooral MongoDB minder goed. Dit verschil is te verklaren: omdat bij deze testcasus ook de accountgegevens opgehaald worden en de accountgegevens bij de relationele DBMS-en in een aparte tabel worden opgeslagen, is er een SQL *Join* nodig. De *Join* zorgt ervoor dat de personen met bijbehorende accountgegevens worden geretourneerd. Terwijl de *Join* voor een vertraging zorgt bij MySQL, Derby en Oracle, hebben MongoDB en CouchDB hier geen last van. Beide ondersteunen *embedded documents*: accountgegevens worden in hetzelfde document (rij) opgeslagen als de persoonsgegevens, waardoor geen *Join*-achtige constructie nodig is.



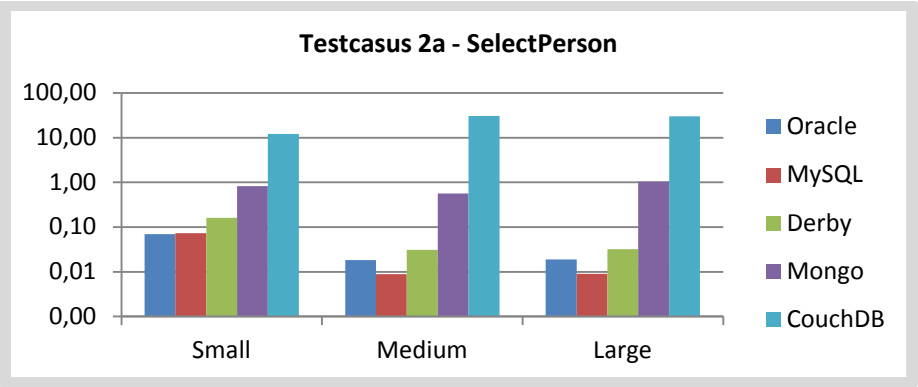
Ten opzichte van testcasus 1b worden naast de accountgegevens nu ook alle onderlinge connecties (relaties) tussen de personen opgehaald. Voor de relationele DBMS-en (Oracle, MySQL en Derby) betekent dit, dat er een tweede *Join* nodig is voor het ophalen van de bijbehorende connecties. De tweede *Join* verklaart de verschuiving in de testresultaten: terwijl voor MongoDB en CouchDB (door de *embedded* documenten) de resultaten ten opzichte van testcasus 1b nagenoeg gelijk blijven, zijn de prestaties van de relationele DBMS-en flink gedaald. Hoe meer *joins,* hoe slechter de prestaties van de relationele DBMS-en ten opzichte van de *document-oriented* DBMS-en.

6.8.2 Testcasus 2 – Een enkel persoon ophalen

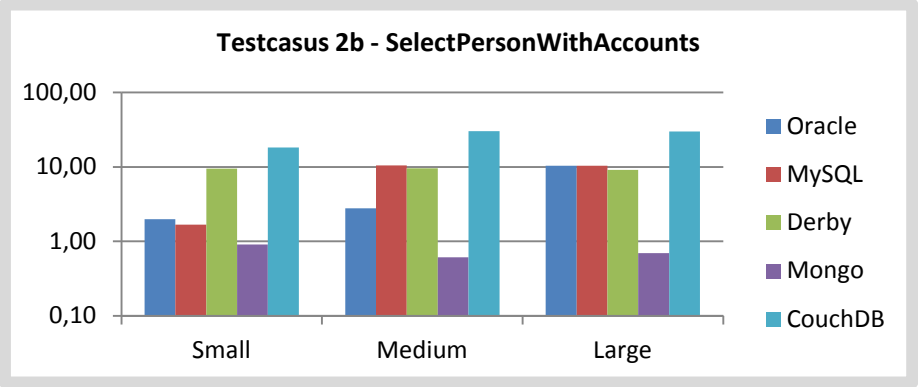
Deze sectie beschrijft de testresultaten van het ophalen van een enkel persoon uit de database. De test is opgedeeld in de drie volgende concrete testcasussen:

1a: Het ophalen van een enkel persoon.
1b: Het ophalen van een enkel persoon met bijbehorende accounts.
1c: Het ophalen van een enkel persoon met bijbehorende accounts en bijbehorende connecties.

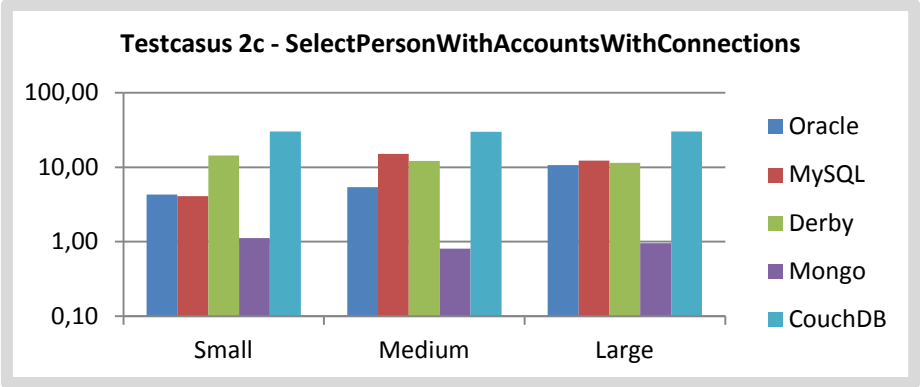
Onderstaande grafieken geven de testresultaten weer voor deze drie testcasussen.



In vergelijking met testcasus 1a (zelfde test, alleen meer personen tegelijk), zijn de resultaten van de relationele databasemanagementsystemen nagenoeg gelijk gebleven, terwijl MongoDB en CouchDB ongeveer tien keer slechter presteren. MySQL komt als beste uit de test.



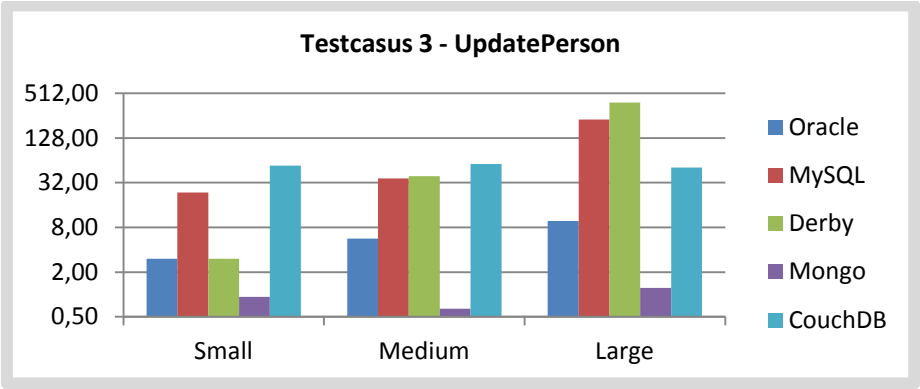
MongoDB is het best presterende DBMS van deze testcasus. Ten opzichte van testcasus 2a worden bij deze test ook de bijbehorende accountgegevens van een persoon opgehaald. Bij de relationele DBMS-en is hierdoor een tijdrovende SQL join nodig in tegenstelling tot MongoDB waar de accountgegevens in hetzelfde document staan. Van de relationele DBMS-en presteert Oracle het beste en Derby het slechtst.



In de lijn van verwachting (zie ook testcasus 1b en 1c) verandert er weinig aan de testresultaten van de document-oriented databasemanagementsystemen. De testresultaten van de relationele DBMS-en zijn allen in verhouding (zie testcasus 2b) door een extra benodigde SQL join trager geworden. Van de relationele DBMS-en presteert Oracle het beste en Derby het slechtst.

6.8.3 Testcasus 3 – Updaten van een persoon

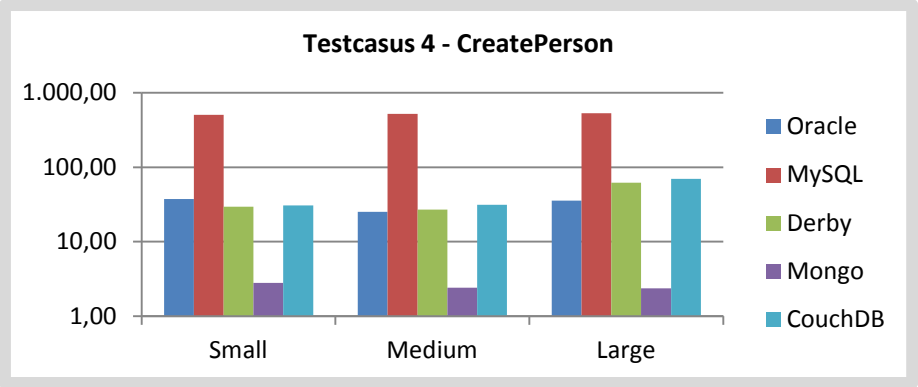
Deze testcasus meet de tijd voor het updaten van de naam van een enkel persoon in de database. Onderstaande grafiek geeft een overzicht van de testresultaten van deze testcasus.



De grote winnaar van deze test is MongoDB. Waar MongoDB ongeveer 1ms over het updaten van een persoon doet, duurt het updaten bij bijvoorbeeld MySQL, gemiddeld over de datasets, ongeveer 80ms. Wat opvalt bij de resultaten is dat de document-oriented DBMS-en geen noemenswaardige verschillen tussen de grootte van datasets vertonen, terwijl er bij de relationele DBMS-en een lineaire groei is te ontdekken naarmate de database (tabel) groeit qua inhoud.

6.8.4 Testcasus 4 – Toevoegen van een persoon

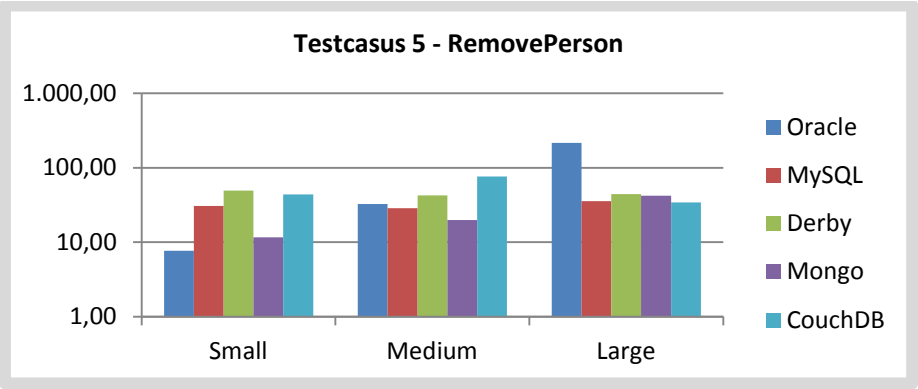
Deze testcasus meet de tijd voor het toevoegen van een enkel persoon met bijbehorende accounts en connecties in de database. Onderstaande grafiek geeft een overzicht van de testresultaten van deze testcasus.



MongoDB is de grote winnaar van deze test en laat de overige DBMS-en op grote achterstand. MySQL presteert het slechts: het toevoegen van één persoon duurt 500ms. Van de relationele DBMS-en presteert Oracle het best (ongeveer 30ms).

6.8.5 Testcasus 5 – Verwijderen van een persoon

Deze testcasus meet de tijd voor het verwijderen van een enkel persoon met bijbehorende accounts en connecties uit de database. Onderstaande grafiek geeft een overzicht van de testresultaten van deze testcasus.



Wat opvalt bij het bekijken van de grafiek is de lineaire groei van MongoDB en Oracle naarmate de dataset groeit. Bij deze testcasus is niet één grote winnaar of verliezer aan te wijzen voor alle datasets, maar als we per dataset kijken, kunnen de volgende conclusies getrokken worden:

- **Small**: Oracle en Derby zijn respectievelijk het best en het slechtst presterende DBMS.
- **Medium**: MongoDB en CouchDB zijn respectievelijk het best en het slechtst presterende DBMS.
- **Large** dataset: CouchDB en Oracle zijn respectievelijk het best en het slechtst presterende DBMS.

6.8.6 Evaluatie

Om een goed algemeen overzicht te krijgen van verhoudingen tussen de DBMS-en is onderstaande evaluatietabel opgesteld (Tabel 4). Bij elke testcasus is voor elk DBMS één tot en met vier sterren toegekend in verhouding met het testresultaat waarbij vier sterren het hoogst haalbaar is.

	1a	1b	1c	2a	2b	2c	3	4	5	Totaal
Derby	★★	★★	★	★★	★	★	0	★★	★★	15
MySQL	★★★★	★★★★	★★	★★★★	★★	★	0	0	★★	19
Oracle	★★★	★★★	★★	★★★	★★★	★★	★★	★★	★★	22
MongoDB	★	★★★	★★★★	★	★★★★	★★★★	★★★★	★★★★	★★★	28
CouchDB	0	0	0	0	0	0	★	★★	★★	5

Tabel 4 – Beoordeling DBMS-en op basis van de testresultaten

MongoDB: is de grote winnaar van de performancetest: zes van de negen testcasussen komt MongoDB als beste uit de test waaronder de twee casussen met de zwaarste weging (1c en 2c). Wanneer de informatie echter in één tabel past (testcasus 1a en 2a), is een relationeel DBMS (met name MySQL) een betere keuze. Voor optimalisatiemogelijkheden van het DBMS wijdt de officiële website een speciale pagina met tips om de performance verder te verbeteren (137).

2. Oracle: scoort voor elke testcasus een voldoende en is hierdoor het meest stabiele DBMS qua testresultaten. Zonder in één testcasus als beste te presteren, eindigt Oracle - mede door de tegenvallende schrijfperformance van MySQL – op de tweede plaats. Oracle is echter wel het best presterende relationele DBMS bij de zwaarst wegende testcasussen (1c en 2c). Door de vele instellingsmogelijkheden kan Oracle voor bijna elke denkbare situatie geoptimaliseerd worden (111).

3. MySQL: is de beste keuze wanneer het gaat om het ophalen van meerdere rijen in één keer. Wanneer de informatie echter over meer dan twee tabellen verspreid is, is MongoDB de betere keuze. Grote tegenvaller is de schrijf-performance (*insert* en *update* queries) van MySQL. Wel zijn er nog optimalisatiemogelijkheden om de schrijfperformance te verbeteren (112).

4. Derby: is het slechtst presterende relationele databasemanagementsysteem en presteert op geen enkel onderdeel boven gemiddeld. Wel is een uitgebreid document beschikbaar met optimalisatietips wat de performance nog zou kunnen verbeteren (113).

5. CouchDB: is in zes van de testcasussen het slechtst presterende DBMS. Het grote verschil kan deels verklaard worden: MongoDB schrijft eerst naar RAM terwijl CouchDB naar schijf schrijft, zodat *durability (de d in ACID* (114)) gegarandeerd kan worden. Bovendien ondersteunt CouchDB meerderde versies (revisies) van een document (MVCC) wat invloed heeft op de performance.

Aangezien alleen drie relationele en twee document-oriented databasemanagementsystemen op performance getest zijn, toont het resultaat van de performancetest alleen welk relationeel of

document-oriented databasemanagementsysteem — het best presteert. Gezien de, door sociale netwerken gebruikte, *column-oriented* DBMS-en Hbase, Cassandra en Voldemort niet getest zijn, is het onduidelijk of MongoDB ook beter presteert dan deze databasemanagementsystemen. Bij de conclusie van het gehele onderzoek dient hier rekening mee gehouden te worden.

Zie hoofdstuk 7 voor de algehele conclusie: het bepalen van het meest geschikte DBMS voor gebruik in het SNScanner project.

7 Conclusie en aanbevelingen

De te beantwoorden hoofdvraag tijdens dit onderzoek was:

- Wat is het meest geschikte DBMS voor de massa opslag van onderling gekoppelde, sociale data?

Het antwoord op deze hoofdvraag luidt:

> **Tijdens het uitgevoerde onderzoek is MongoDB als meest geschikte database-managementsysteem naar voren gekomen voor gebruik in het Social Network Scanner project. Tijdens het onderzoek is gebleken dat MongoDB aan de gestelde criteria van het Social Network Scanner project voldoet en als beste presteerde tijdens de performancetest.**

Een kanttekening bij de performancetest is dat de *column-oriented* databasemanagementsystemen Hbase, Cassandra en Voldemort niet getest konden worden. Maar omdat MongoDB aan alle gestelde eisen voldoet én als best presteerde tijdens de performancetest, is MongoDB de meest geschikte keuze voor gebruik in het Social Network Scanner project.

Aanbevelingen en uitbreidingen op dit onderzoek:

1. Voor toekomstige projecten kan het gebruik van een *column-oriented* databasemanagement-systeem gewenst zijn. Gezien deze DBMS-en nog niet getest zijn tijdens de performancetest, kan het voor toekomstige projecten nuttig zijn dit alsnog te doen.

2. Verder zijn er nog vele optimalisatiemogelijkheden voor elk van de databasemanagement-systemen. Bij toekomstig onderzoek kan het interessant zijn, om te onderzoeken met welk geoptimaliseerd databasemanagementsysteem de meeste performancewinst behaald kan worden.

MongoDB *is het meest geschikte DBMS voor de massa opslag van onderling gekoppelde, sociale data.*

8 Verklarende woordenlijst

9 Bibliografie

Gezien de verklarende woordenlijst en bibliografie overeenkomen met die van de scriptie, is besloten om deze niet in het afstudeerdossier bij te sluiten.