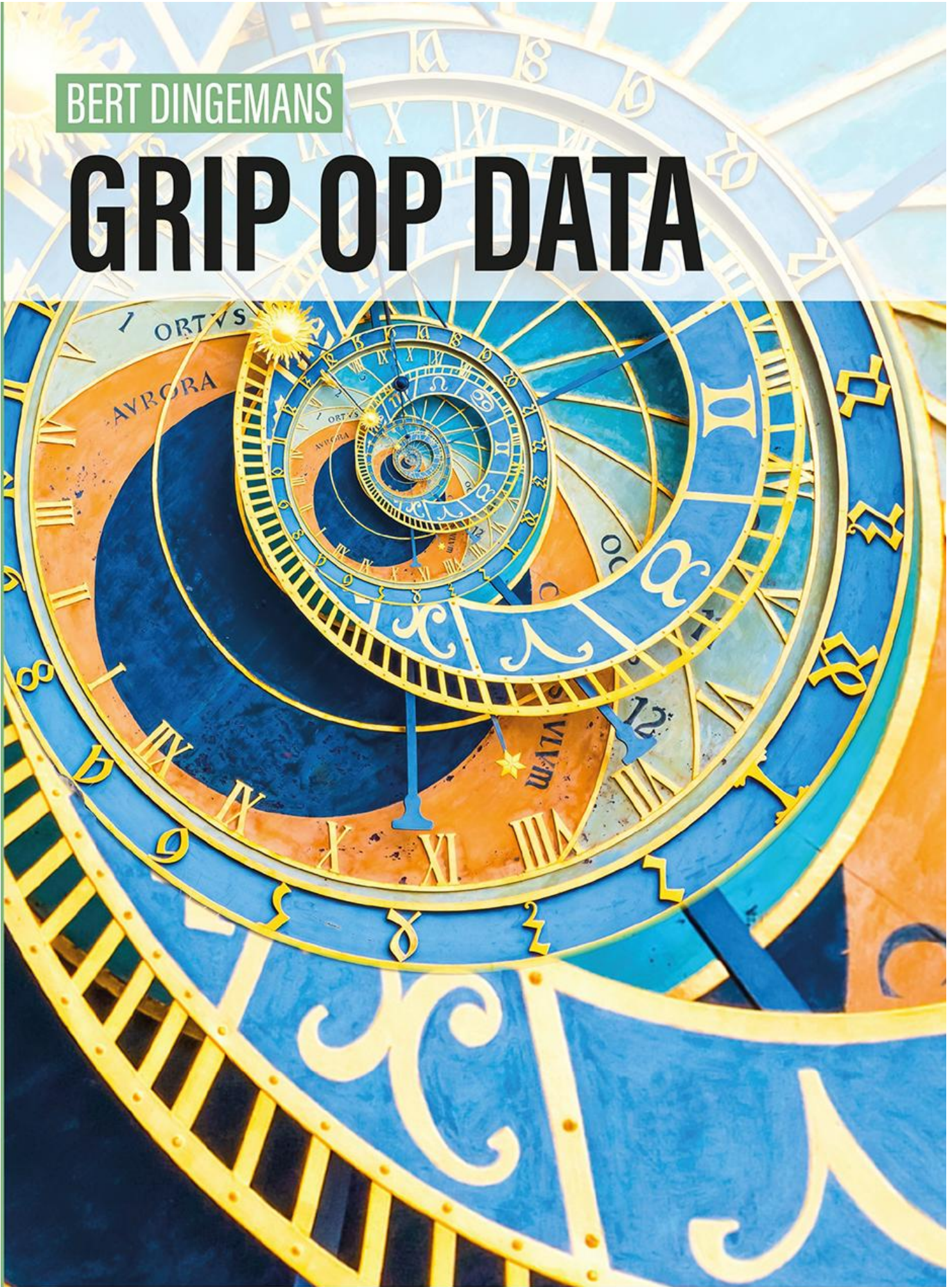


BERT DINGEMANS

GRIP OP DATA



Archigraph

Een ArchiMate model in een grafen database



Ir. Ing. Bert Dingemans

bert@data-docent.nl

INHOUDSOPGAVE

Inhoudsopgave	3
Inleiding	4
ArchiMate als Graph	4
Graph in SQL-Server	5
Graph datamodellen	6
Stereotype als Node tabel.....	6
Tabel gebaseerde queries	7
Stereotype als kolom in een Node tabel	9
Kolom gebaseerde queries	10
Grafen in PowerBI.....	11
Tot slot	13
Over de auteur	14

Inleiding

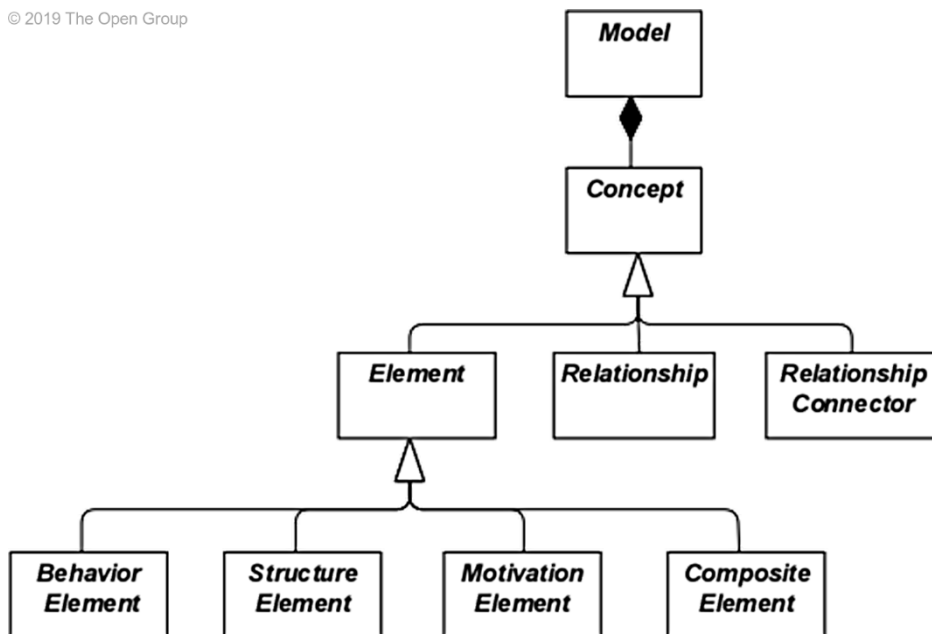
In dit whitepaper wordt een ArchiMate model omgezet naar een grafen database in SQL-Server. Een grafen database is een nieuwe datastructuur voor de opslag van elementen die via connectoren met elkaar verbonden zijn. In dit whitepaper gaan we onderzoeken of een grafen database het leven van de architectuurmodeller eenvoudiger maken bij het bevragen van een ArchiMate model. Dat doen we door twee vormen van modelleren te verkennen. Beiden zijn een grafen database waarbij in de ene variant ieder ArchiMate concept een eigen tabeldefinitie krijgt. In het andere model is dat er één type element wordt gebruikt maar dat er een eigenschap is opgenomen voor de stereotypen.

Dit whitepaper is een onderdeel van meerdere whitepapers over databases en modelleren. Databases zijn onder andere gericht op relationele databases en NoSQL databases. We gaan in op het opvragen van data op basis van het select statement binnen de structured query language op basis van data opgeslagen in de grafen database. Daarnaast zijn er whitepapers over datamodelleren in deze serie van whitepapers, zie <https://data-docent.nl>.

ArchiMate als Graaf

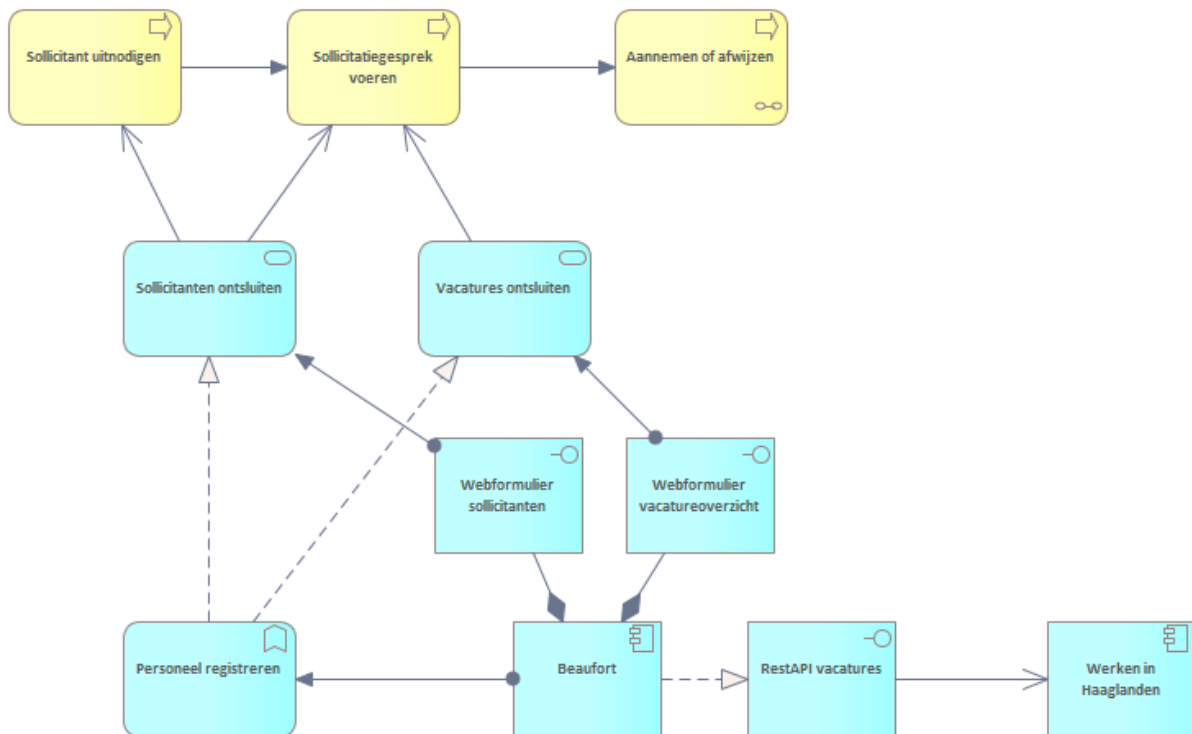
Het metamodel van ArchiMate kenmerkt zich dat het in de basis een structuur is die is opgebouwd uit twee entiteiten namelijk elementen en relaties. Vervolgens hebben alle elementen en relaties stereotypen waarmee het onderscheid in het model gemaakt kan worden. Hieronder zie je een voorbeeld dat een klein deel van dit metamodel weergeeft

© 2019 The Open Group



Dit metamodel geeft de structuur weer naar de verschillende typen en vervolgens op basis van een specialisatiestructuur worden de stereotypen bepaald qua vormgeving en qua regels rond de concepten. Vervolgens wordt in het metamodel bepaald dat twee elementen via relaties met elkaar kunnen worden verbonden. Ook deze relaties kunnen weer een eigen stereotype hebben qua betekenis en vormgeving.

Hieronder zie je een eenvoudig model terug van hoe elementen en relaties in ArchiMate eruit zien op basis van bovengenoemd metamodel



In het diagram zie je in de elementen in de rechterbovenhoek een symbooltje staan wat het element een bepaalde betekenis geeft, de weergave van het stereotype. Aan de connectoren zie je dat er meerdere soorten relaties zijn met allen een eigen betekenis en dus stereotype. Dit is kenmerkend voor het ArchiMate model en neigt daarmee sterk naar opslag in een grafendatabase.

In Sparx Enterprise Architect wordt het model zoals opgesteld opgeslagen in een relationele database waarbij er twee tabellen in de database heel erg belangrijk zijn. Namelijk de tabel `t_object` en `t_connector`. In `t_connector` zijn er twee foreign keys opgenomen die afdwingen dat een connector gekoppeld is aan twee elementen (in de `t_object` tabel).

Dit geeft in basis meer dan voldoende mogelijkheden om queries te maken. Maar de queries worden al snel complex met vele joins. Is een grafendatabase een verbetering? Dat is de vraag die we in dit whitepaper willen beantwoorden.

Graph in SQL-Server

SQL-Server heeft een mogelijkheid om eenvoudige grafen inrichting te implementeren. Je maakt tabellen aan en relaties die op een graaf wijze aan elkaar gerelateerd kunnen worden. Deze graaf entiteiten hebben een bepaalde opbouw die je kunt gebruiken om graaf queries te kunnen maken.

In de IDEA AddOn heb ik een aantal scripts gemaakt die het mogelijk maken om de structuur voor grafen modellen te genereren op basis van je repository inhoud.

Hieronder een voorbeeld script voor het genereren van een grafentabel structuur. In dit geval op basis van de ArchiMate stereotypes die in de repository zijn gebruikt zoals onderstaande twee entiteiten create

```
CREATE TABLE A_ApplicationComponent (  
  ID INTEGER PRIMARY KEY,  
  [name] VARCHAR(100),  
  [note] text,  
  [version] VARCHAR(100),  
  [synonyms] VARCHAR(100)  
) AS NODE;
```

```
CREATE TABLE A_ApplicationFunction (  
  ID INTEGER PRIMARY KEY,  
  [name] VARCHAR(100),  
  [note] text,  
  [version] VARCHAR(100),  
  [synonyms] VARCHAR(100)  
) AS NODE;
```

Verder een voorbeeldscript voor het definiëren van een edge op basis van een create edge script voor een ArchiMate access relationship

```
CREATE TABLE A_Access (  
  name VARCHAR(100),  
  version VARCHAR(100),  
  note text  
) AS EDGE;
```

Interessant om te zien is dat de twee entiteiten worden gebaseerd op een Node en een Edge entiteit. Dat is een mooie opzet die je ook terugziet in OrientDB en Neo4J. Twee grafendatabases die veel gebruikt worden voor het inzetten van grafendata. SQL-Server is een vereenvoudigde uitwerking hiervan.














































Graph datamodelen

In dit whitepaper zijn twee scenario's uitgewerkt:

- **Stereotype als Node tabel:** De eerste aanpak is dat voor ieder ArchiMate stereotype als een tabel wordt uitgewerkt.
- **Stereotype als kolom in een Node tabel:** Tweede aanpak is dat er één Node gemaakt wordt voor de ArchiMate elementen en één Edge voor de ArchiMate relaties. Vervolgens wordt het stereotype in de kolom weggeschreven

STEREOTYPE ALS NODE TABEL

In onderstaande afbeelding zie je welke elementen als Node zijn aangemaakt en welke als Edge zijn aangemaakt. Dit is gegenereerd op basis van een script. Op de web data-docent staan de genereerscripts voor de create en insert scripts hiervoor (zie onderaan het whitepaper).

- [-] Graph Tables
 - [+]  dbo.A_ApplicationComponent
 - [+]  dbo.A_ApplicationFunction
 - [+]  dbo.A_ApplicationInterface
 - [+]  dbo.A_ApplicationService
 - [+]  dbo.A_BusinessActor
 - [+]  dbo.A_BusinessEvent
 - [+]  dbo.A_BusinessObject
 - [+]  dbo.A_BusinessProcess
 - [+]  dbo.A_BusinessRole
 - [+]  dbo.A_Constraint
 - [+]  dbo.A_DataObject
 - [+]  dbo.A_Deliverable
 - [+]  dbo.A_DistributionNetwork
 - [+]  dbo.A_Driver
 - [+]  dbo.A_Equipment
 - [+]  dbo.A_Facility
 - [+]  dbo.A_Gap
 - [+]  dbo.A_Goal
 - [+]  dbo.A_Junction
 - [+]  dbo.A_Material
 - [+]  dbo.A_Node
 - [+]  dbo.A_Outcome
 - [+]  dbo.A_Plateau
 - [+]  dbo.A_Principle
 - [+]  dbo.A_Requirement
 - [+]  dbo.A_Stakeholder
 - [+]  dbo.A_SystemSoftware
 - [+]  dbo.A_TechnologyFunction
 - [+]  dbo.A_TechnologyInterface
 - [+]  dbo.A_TechnologyProcess
 - [+]  dbo.A_TechnologyService
 - [+]  dbo.A_WorkPackage
 - [+]  dbo.A_Element
 - [+]  dbo.A_Access
 - [+]  dbo.A_Aggregation
 - [+]  dbo.A_Assignment
 - [+]  dbo.A_Association
 - [+]  dbo.A_Composition
 - [+]  dbo.A_Flow
 - [+]  dbo.A_Influence
 - [+]  dbo.A_Realization
 - [+]  dbo.A_Serving
 - [+]  dbo.A_Specialization
 - [+]  dbo.A_Triggering
 - [+]  dbo.A_Relation

TABEL GEBASEERDE QUERIES

In de queries hieronder zien we een aantal voorbeelden van queries die op basis van deze opzet mogelijk zijn. De queries worden kort toegelicht bij de voorbeelden. Bij het werken met grafenqueries in SQL-Server moet er gebruik gemaakt worden van een door Microsoft ontwikkelde stored function die de query afhandeling en het leggen van joins op eenvoudiger wijze mogelijk maakt.

```
SELECT AC.name as component, AF.name as afunction
FROM A_ApplicationComponent AS AC
, A_Assignment
, A_ApplicationFunction AS AF
WHERE MATCH (AC-(A_Assignment)->AF)
AND af.name NOT LIKE 'Web%'
```

Op basis van deze query zie je dat de assignment relaties tussen applicatie functies en applicatie componenten worden getoond. De resultaatset hieronder

component	afunction
Formdesk	Work flow
Beaufort	Personeelsinformatiesysteem
Word	Correspondentie en mail vervaardigen en archiveren
Outlook	Correspondentie en mail vervaardigen en archiveren
Beaufort	Personeel registreren

```
SELECT AC.name as appcomponent, AF.name as appfunction, S.name as appservice
FROM A_ApplicationComponent AS AC
, A_Assignment
, A_ApplicationFunction AS AF
, A_Realization
, A_ApplicationService as S
WHERE MATCH (AC-(A_Assignment)->AF-(A_Realization)->S )
```

In deze opzet een extra uitwerking in het model waarbij je een deelmodel toont van applicatie component, function en service met elkaar in relatie brengt.

```
SELECT AC.name as appcomponent
, AF.name as appfunction
, S.name as appservice
, BP.name as BusProcess
FROM A_ApplicationComponent AS AC
, A_Assignment
, A_ApplicationFunction AS AF
, A_Realization
, A_ApplicationService as S
, A_Serving
, A_BusinessProcess as BP
WHERE MATCH (AC-(A_Assignment)->AF-(A_Realization)->S-(A_Serving)->BP )
```

Verdere uitbreiding van de query nu met het koppelen van een deel van de applicatielaag aan de bedrijfsprocessen.

```
SELECT BE.name as BussProces1
, BP2.name as BussProces2
, 'Event-Process' as type
FROM A_BusinessEvent AS BE
, A_Triggering
```



```

, A_BusinessProcess AS BP2
WHERE MATCH(BE-(A_Triggering)->BP2)
UNION
SELECT BP1.name as BussProces1
, BP2.name as BussProces2
, 'Process-Process' as type
FROM A_BusinessProcess AS BP1
, A_Triggering
, A_BusinessProcess AS BP2
, A_BusinessEvent as BE
WHERE MATCH(BP1-(A_Triggering)->BP2)


```


Wil je in de resultaatset ook het stereotype opnemen dan kan dat je moet een constante opnemen waarmee je weergeeft wat het stereotype is op basis van de tabelstructuur.

De bovenstaande queries laten zien dat de indeling op basis van de op stereotype gebaseerde tabellen het mogelijk maken om specifieke queries te maken. Dat is in een aantal situaties een voordeel. Er is wel een nadeel als je kijkt naar het metamodel van ArchiMate. Er zijn aspecten en binnen aspecten kun je meerdere stereotypes terugvinden. Wil je relaties gaan leggen om inzichtelijk maken hoe de verschillende aspecten aan elkaar gerelateerd zijn dan wordt de query al snel zeer complex. In het volgende scenario gaan we hier nader op in om dit inzichtelijk te maken.

STEREOTYPE ALS KOLOM IN EEN NODE TABEL

In onderstaande afbeelding zie je de andere opzet waarbij de edge en node tabellen worden getoond. Er zijn een aantal kolommen gedefinieerd die de vulling van de ArchiMate concepten laat zien. Je ziet hier duidelijk het Sparx datamodel terug. Daarnaast zie je een aantal extra kolommen die afkomstig zijn door de overerving van de Node en de Edge tabellen die nodig zijn om de grafen queries mogelijk te maken.

A_Element	
	graph_id_5847D98A20BC4D05B...
	[\$node_id_DA943D8806C945C1...
	ID
	name
	stereotype
	note
	version
	synonyms

A_Relation	
	graph_id_EDE0A121BF6848938...
	[\$edge_id_8787C7540B8B4CB7...
	from_obj_id_3AD503554865415...
	from_id_7696A6261D374A1A8C...
	[\$from_id_24C0DD830B01465D...
	to_obj_id_18DC27C58D724ED9...
	to_id_6723044349A94D6E84433...
	[\$to_id_1BF1CEDAB3D3489EAB...
	ID
	name
	stereotype
	note

KOLOM GEBASEERDE QUERIES

In onderstaande queries zie je dat het stereotype door de insert statements worden gevuld in de kolom die daarvoor is aangemaakt. Vervolgens zie je dat ook in de Edges tabel een kolom stereotype is opgenomen. Hiermee kun je queries combineren op basis van het grafen model en op basis van sql opbouw in het statement. Ik weet niet of dit gebaseerd is mijn voorkeur voor SQL of dat dit een aanpak is die duidelijke voordelen heeft ten opzichte van de vorige opzet.

```
SELECT S.name as source
, S.stereotype as stype
, T.name as target
, T.stereotype as ttype
, C.stereotype as connection
FROM A_Element AS S
, A_Relation as C
, A_element as T
WHERE MATCH (T-(C)->S)
AND c.stereotype = 'Realization'
```

In bovenstaande query zie je dat je alle nodes met behulp van de edge entiteiten aan elkaar verbindt en vervolgens op basis van de stereotype in de kolommen gaat filteren wat je precies wilt zien. In bovenstaand voorbeeld met een realisatie stereotype van de edge.

```
SELECT S.name as source
, S.stereotype as stype
, T.name as target
, T.stereotype as ttype
, C.stereotype as connection
FROM A_Element AS S
, A_Relation as C
, A_element as T
WHERE MATCH (T-(C)->S)
AND 'ApplicationComponent' IN(S.stereotype, t.stereotype)
```

Bovenstaande query laat zien hoe je alle elementen gerelateerd aan een Applicatiecomponent element in de bron of de doel node.

```
SELECT S.name as source
, S.stereotype as stype
, I.name as intermediate
, I.stereotype as itype
, T.name as target
, T.stereotype as ttype
FROM A_Element AS S
, A_Relation as CI
, A_Element AS I
, A_Relation as IT
, A_element as T
WHERE MATCH (T-(CI)->I-(IT)->S)
```

source	stype	intermediate	itype	target	ttype
Sollicitant	Stakeholder	Tevreden medewerker	Principle	KLantonderzoek doen	Constraint
Aannemen of afwijzen	BusinessProcess	Sollicitatiegesprek voeren	BusinessProcess	Vestigingsmanager	BusinessActor
Besluit	BusinessObject	Sollicitant	BusinessObject	Personeelsinfo	DataObject

Correspondentie	DataObject	Correspondentie en mail vervaardigen en archiveren	ApplicationFunction	Outlook	ApplicationComponent
Aannemen of afwijzen	BusinessProcess	Correspondentie en mail vervaardigen en archiveren	ApplicationFunction	Outlook	ApplicationComponent
Vacatures ontsluiten	ApplicationService	Webformulier vacatureoverzicht	ApplicationInterface	Beaufort	ApplicationComponent
Werken in Haaglanden	ApplicationComponent	RestAPI vacatures	ApplicationInterface	Beaufort	ApplicationComponent
Sollicitanten ontsluiten	ApplicationService	Webformulier sollicitanten	ApplicationInterface	Beaufort	ApplicationComponent
Vacatures ontsluiten	ApplicationService	Personeel registreren	ApplicationFunction	Beaufort	ApplicationComponent
Personeelsinfo	DataObject	Personeel registreren	ApplicationFunction	Beaufort	ApplicationComponent
Sollicitatiegesprek voeren	BusinessProcess	Personeel registreren	ApplicationFunction	Beaufort	ApplicationComponent
Sollicitanten ontsluiten	ApplicationService	Personeel registreren	ApplicationFunction	Beaufort	ApplicationComponent
Correspondentie	DataObject	Correspondentie en mail vervaardigen en archiveren	ApplicationFunction	Word	ApplicationComponent

Bovenstaande queries laten zien dat het bevragen van een ArchiMate model in een grafendatabase duidelijk eenvoudiger wordt. Dit maakt het mogelijk om krachtige analyses te maken. In de voorbeelden is te zien hoe dit in te zetten is. Wel zie je dat je vooraf moet definiëren welke entiteiten terugkomen in het statement.

Op basis hiervan kun je op eenvoudige wijze onderzoeken doen in je ArchiMate model. Bijkomend voordeel is dat alles in één database gebeurt. Je zou ervoor kunnen zorgen dat als er iets wijzigt in het Sparx Enterprise ArchiMate model een trigger afgaat die vervolgens de grafen tabellen bijwerkt.

Grafen in PowerBI

In Sparx Enterprise Architect kun je vanzelfsprekend visualisaties maken van de ArchiMate model in feitelijk een grafenweergave. Echter deze weergave is gebaseerd op het relationele model in de Sparx repository. Echter willen we de grafendatabase visualiseren in de node en edge tabellen dan kan dat niet direct. Hiervoor zijn interessante mogelijkheden in PowerBI.

PowerBI kent een aantal visuele elementen in de dashboards waarmee je de graaf kunt visualiseren. Hiertoe moet je een aantal zaken combineren. De eerste is dat er een view dient te komen die de nodes en edges op basis van een view weer toont in een tabelweergave. Dat kan wederom met een select statement naar de nodes en edges. Vervolgens wordt de inhoud van onderstaande view geïmporteerd in PowerBI (of gekoppeld).

Interessant in de view is dat we een combinatie gaan maken met een tabel met metadata voor ArchiMate over de Layers en Aspecten zoals die voorkomen in het metamodel. Een mooie weergave van de kracht van grafendatabases in combinatie met data die in tabellen is opgeslagen en met joins worden gerelateerd.

```
CREATE OR ALTER view [dbo].[ArchiGraphView]
as
```

```

select src.name as srcname
, replace(src.stereotype, 'ArchiMate_', '') as srcstereotype
, smm.Layer as srclayer
, smm.[Column] as srcaspect
, replace(con.stereotype, 'ArchiMate_', '') as constereotype
, trg.name as trgname
, replace(trg.stereotype, 'ArchiMate_', '') as trgstereotype
, tmm.Layer as trglayer
, tmm.[Column] as trgaspect
from t_object as src
inner join t_connector as con on con.Start_Object_ID = src.Object_ID
inner join t_object as trg on con.End_Object_ID = trg.Object_ID
left join IDEA_METAMODEL as smm ON src.stereotype = smm.StereoType
left join IDEA_METAMODEL as tmm ON trg.stereotype = tmm.StereoType
where src.stereotype LIKE 'ArchiMate%'
or trg.stereotype LIKE 'ArchiMate%'

```

Onderstaande afbeeldingen van de visuals in PowerBI laten zien hoe je op eenvoudige wijze de de grafen zichtbaar kunt maken op verschillende niveaus in het ArchiMate metamodel. In de eerste afbeelding op basis van de lagen en aspecten. Daaronder op basis van de stereotypen



Over de auteur



Bert Dingemans is trainer op het vlak van data architectuur, data management en Big Data. Hij heeft een passie voor modelleren, modelleertools en het effectief inzetten van geautomatiseerde hulpmiddelen om modellen effectief in te zetten in de praktijk. Meer whitepapers zijn te vinden op <https://data-docent.nl>. Bert is per mail te

bereiken via bert@data-docent.nl