Een eenvoudige model van een boomstructuur

De basis van een hiërarchische query is een simpele verwijzing, van de ene rij naar een andere rij van een tabel. Hieronder ziet u een tabel WERKNEMERS (je kan deze downloaden via werknemers.txt), waarbij de kolom MGR verwijst naar een kolom PERSNR van een andere rij.

PERSNR	NAAM	FUNCTIE	MGR	SAL	TOESLAG	KANTNR
3381	SMITS	KLERK	7902	2400		20
3462	ALKEMA	VERKOPER	4621	2600	300	30
3518	WALSTRA	VERKOPER	4621	2250	500	30
3930	PIETERS	MANAGER	6221	3975		20
4510	VERGEER	VERKOPER	4621	2250	1400	30
4621	KLAASEN	MANAGER	6221	3850		30
5810	HEUVEL	MANAGER	6221	3450		10
5931	SANDERS	ANALIST	3930	4000		20
6221	KRAAY	DIRECTEUR		6000		10
6500	DROST	VERKOPER	4621	2500	0	30
6681	ADELAAR	KLERK	5931	2100		20
7900	APPEL	KLERK	4621	1950		30
7902	VERMEULEN	ANALIST	3930	3900		20
8222	MANDERS	KLERK	5810	2300		10

We zien bijvoorbeeld dat SMITS een MGR heeft van 7902. Dit nummer zien we terug als PERSNR van VERMEULEN. Vermeulen is dus de manager van Smits. Op zijn beurt heeft Vermeulen een MGR van 3930, en dat is het PERSNR van Pieters. En Pieters blijkt als manager Kraay te hebben. Kraay heeft in de kolom MGR geen waarde staan: hij heeft geen manager, want hij is de directeur.

Veel implementaties van boomstructuren gaan uit van een enkel knooppunt (of node), waaronder diverse child-nodes voor kunnen komen, die elk weer hun eigen child nodes kunnen hebben. De bovenste node wordt dan de root-node genoemd.

Bovenstaande tabelstructuur is eenvoudiger opgebouwd, maar met hetzelfde effect: in plaats van een verwijzing naar mogelijk meerdere child-nodes bevat elke node hooguit één verwijzing naar een bovenliggende node: de parent node. De gehele boomstructuur kan uit die parent-child-verwijzingen worden afgeleid. Het lastige is alleen dat die boomstructuur nu nog moeilijk te zien is.

In de casus zullen we ook uitgaan van een verzameling nodes met verwijzingen naar parent-nodes. In volgende paragrafen wordt uitgelegd hoe bij het doorlopen van de rijen van een tabel informatie over de onderliggende boomstructuur kan worden getoond. In de casus zullen we die functionaliteit overbrengen naar een gespecialiseerde Iterator, die bedoeld is om een verzameling node objecten met hun parent nodes te doorlopen.

De boomstructuur in volgorde

Om de onderliggende boomstructuur in een tabel goed te kunnen zien, kan binnen Oracle een hiërarchische query worden gebruikt. De basis daarvan zijn CONNECT BY PRIOR en START WITH. Met START WITH kiezen we de rij waarmee we beginnen, CONNECT BY PRIOR geeft aan op welke manier de volgende rij wordt gekozen.

```
select PERSNR, MGR, NAAM
from werknemers
start with MGR  is null
connect by MGR = prior PERSNR;
```

Hier kiezen we dus om te beginnen met de rij waarvan de kolom MGR leeg is (is null). Een volgende rij wordt gekozen, waarvoor geldt dat MGR gelijk is aan het vorige (prior) PERSNR, met dit als resultaat:

PERSNR	MGR	NAAM
6221		KRAAY
3930	6221	PIETERS
5931	3930	SANDERS
6681	5931	ADELAAR
7902	3930	VERMEULEN
3381	7902	SMITS
4621	6221	KLAASEN
3462	4621	ALKEMA
3518	4621	WALSTRA
4510	4621	VERGEER
6500	4621	DROST
7900	4621	APPEL
5810	6221	HEUVEL
8222	5810	MANDERS

De boomstructuur in volgorde

De boomstructuur is nu nog niet helemaal duidelijk, maar aan de volgorde is al wel te zien dat Pieters onder Kraay valt, Sanders onder Pieters, en Adelaar onder Sanders. Adelaar heeft niemand onder zich: de volgende rij is Vermeulen: dit is na Sanders de tweede werknemer onder Pieters.

Als we alleen iedereen onder Pieters willen zien, dan passen we de start with clausule aan:

```
select PERSNR, MGR, NAAM
from werknemers
start with NAAM = 'PIETERS'
connect by MGR = prior PERSNR;
```

PERSNR	MGR	NAAM
3930	6221	PIETERS
5931	3930	SANDERS
6681	5931	ADELAAR
7902	3930	VERMEULEN
3381	7902	SMITS

Voor de casus kunnen we hier alvast wat conclusies uit trekken:

- We kunnen een boomstructuur samenstellen uit een **verzameling** nodes die verwijzen naar hun parent-node (List)
- We kunnen de boomstructuur **doorlopen**, beginnend bij een op te geven parent node (Iterator)
- De **volgorde** bij het doorlopen van de boomstructuur wordt bepaald door de hiërarchie (Comparable)

Level

De boomstructuur in de tabel wordt al duidelijker door de logische volgorde vanaf de root, maar de gevonden informatie is nog beperkt. Gelukkig kan er meer informatie worden getoond in een hiërarchische query.

Ten eerste kan per regel het niveau (of level) binnen de hiërarchie worden getoond. In het voorgaande voorbeeld staat Pieters op niveau 1. Daaronder komt Sanders, dus die krijgt niveau 2, et cetera. Dit level wordt een pseudo-kolom genoemd: het staat niet in de tabel, maar wordt bij het doorlopen afgeleid uit de beschikbare informatie:

```
select PERSNR, MGR, NAAM, LEVEL
from werknemers
start with NAAM = 'PIETERS'
connect by MGR = prior PERSNR;
```

PERSNR	MGR	NAAM	LEVEL
3930	6221	PIETERS	**1**
5931	3930	SANDERS	**2**
6681	5931	ADELAAR	**3**
7902	3930	VERMEULEN	**2**
3381	7902	SMITS	**3**

In de casus zullen we ook een level genereren bij het doorlopen van de boomstructuur. Omdat we in Java gewend zijn aan 0-based indexen zal de start node daarin level 0 krijgen, in plaats van level 1. In analogie met bovenstaand voorbeeld: het level van Adelaar kunnen we dan bepalen door in een loop (of recursief) te tellen hoe vaak een parent node kan worden opgehaald, totdat Pieters wordt bereikt.

Connect_by_isleaf

Behalve het niveau, kunnen we zien of een werknemer ook nog manager is van een ander. In de boomstructuur wordt een “blad” (leaf) gezien als een node waaronder geen andere nodes voorkomen. De pseudokolom CONNECT_BY_ISLEAF geeft dat aan met een 1 of een 0, waarbij 1 staat voor true, en 0 voor false.

```
select PERSNR, MGR, NAAM, LEVEL, CONNECT_BY_ISLEAF
from werknemers
start with NAAM = 'PIETERS'
connect by MGR = prior PERSNR;
```

PERSNR	MGR	NAAM	LEVEL	CONNECT_BY_ISLEAF
3930	6221	PIETERS	1	0
5931	3930	SANDERS	2	0
6681	5931	ADELAAR	3	1
7902	3930	VERMEULEN	2	0
3381	7902	SMITS	3	1

We zien nu direct dat Adelaar en Smits geen manager zijn van iemand. Waar het level afhangt van de gekozen start-node geldt dat niet voor connect_by_isleaf.

In de casus maken we ook een isLeaf() methode, maar dan van het type boolean. We zullen deze methode implementeren door per node bij te houden hoe vaak die node als parent node van een andere node wordt aangemerkt. Later zal ook een andere implementatie worden voorgesteld.

Sys_connect_by_path

Het duidelijkste plaatje krijgen we als we ook het pad tonen van de werknemer tot het gekozen begin-niveau. We tonen daarvoor bij elke werknemer en bovenliggende werknemers een gekozen kolomwaarde, en scheiden de niveaus door een gekozen scheidingsteken. In dit voorbeeld gebruiken we de naam als kolomwaarde, en een slash als scheidingsteken.
Hiervoor bestaat de functie sys_connect_by_path:

```
select PERSNR, MGR, NAAM, SYS_CONNECT_BY_PATH(naam,'/')
from werknemers
start with NAAM = 'PIETERS'
connect by MGR = prior PERSNR;
```

PERSNR	MGR	NAAM	SYS_CONNECT_BY_PATH(NAAM,'/')
3930	6221	PIETERS	**/PIETERS**
5931	3930	SANDERS	**/PIETERS/SANDERS**
6681	5931	ADELAAR	**/PIETERS/SANDERS/ADELAAR**
7902	3930	VERMEULEN	**/PIETERS/VERMEULEN**
3381	7902	SMITS	**/PIETERS/VERMEULEN/SMITS**

Merk op dat de volgorde waarin de werknemers getoond worden overeenkomt met de hiërarchie. De boomstructuur wordt zichtbaar gemaakt door onder de “root” (Pieters) van elke “hoofdtak” (Sanders en Vermeulen) eerst alle zijtakken te tonen, voordat door wordt gegaan naar de volgende hoofdtak.

Ook deze functie zullen we vertalen naar een Java-methode. Een uitdaging daarbij is het bepalen welke informatie wordt getoond binnen het pad. Van een kennen we immers alleen de parent. Dat uit een subtype van Node de naam kan worden gebruikt in het pad, is op het niveau van een node nog niet bekend.

Order siblings by

Zoals gezegd wordt bij het doorlopen van de boomstructuur de volgorde bepaald door de hiërarchie, te beginnen bij een gekozen root node. In het voorgaande voorbeeld beginnen we bij Pieters. Te zien is dat er twee nodes direct onder Pieters vallen: Sanders en Vermeulen. Meerdere nodes onder een zelfde parent node worden siblings genoemd (vrij vertaald: broers of zussen). Uit de hiërarchie zelf kan niet worden afgeleid in welke volgorde die siblings moeten worden getoond. Siblings hebben immers een gelijk niveau. Er had ook voor gekozen worden om na Pieters eerst Vermeulen te tonen, met daaronder Smits, en dan pas Sanders, met daaronder Adelaar.

We kunnen de volgorde van de siblings daarom zelf bepalen met order siblings by. In het volgende voorbeeld worden de siblings getoond in volgorde van salaris:

```
select PERSNR, MGR, NAAM, sal, SYS_CONNECT_BY_PATH(naam,'/')
from werknemers
start with NAAM = 'PIETERS'
connect by MGR = prior PERSNR
order siblings by SAL;
```

PERSNR	MGR	NAAM	SAL	SYS_CONNECT_BY_PATH(NAAM,'/')
3930	6221	PIETERS	3975	/PIETERS
7902	3930	VERMEULEN	**3900**	/PIETERS/VERMEULEN
3381	7902	SMITS	2400	/PIETERS/VERMEULEN/SMITS
5931	3930	SANDERS	**4000**	/PIETERS/SANDERS
6681	5931	ADELAAR	2100	/PIETERS/SANDERS/ADELAAR

Omdat Vermeulen minder verdient dan Sanders wordt hij eerst getoond.

Ook in de casus zullen we de volgorde van siblings opgeven.

Welke Java interface kan daarvoor gebruikt worden?