De wereld om ons heen wordt steeds vaker in meer dan twee dimensies gemodelleerd. Geo-informatie in twee dimensies is echter beschikbaar in grote hoeveelheden en op verschillende schalen. Daarnaast is het zo dat deze informatie in veel gevallen nog voldoet en vaak ook de meest efficiënte en meest economische oplossing is. Daarom loont het de moeite te kijken of het mogelijk is huidige geo-databases (DBMSen) die gevuld zijn met 2D geo-informatie uit te breiden met de mogelijkheid om 3D geo-informatie te ondersteunen. Dit levert een systeem op waarin zowel 2D als 3D geo-objecten tezamen met hun niet-ruimtelijke eigenschappen en onderliggende relaties kunnen worden beheerd en bevraagd.

Een oplossing voor een 3D-kadaster?[1)]

drs. J. E. Stoter en prof. dr. ir. P. J. M. van Oosterom, sectie GIS-technologie, TU Delft.

Binnen de sectie GIS-technologie van de TU Delft is een onderzoek gestart naar de integratie van 2D en 3D geo-objecten in één DBMS om zowel de mogelijkheden als de bijkomende complicaties te bestuderen. In dit artikel zal eerst een aantal fundamentele zaken over 3D-modellering en de integratie hiervan in een 2D-omgeving worden behandeld. Oracle Spatial 9i wordt in dit onderzoek gebruikt om de geo-objecten in het DBMS te beheren. We beschrijven ook een voorstel om de huidige datatypen binnen Oracle, die voornamelijk 2D zijn, uit te breiden naar 3D.

Achtergrond van het onderzoek

Dit onderzoek is een onderdeel van het '3D-kadaster', een onderzoek dat in samenwerking met het Kadaster wordt uitgevoerd [1]. Binnen dit onderzoek wordt gekeken op welke wijze het mogelijk is de ruimtelijke component van de rechten bij multifunctioneel ruimtegebruik te registreren. Op deze manier kan het Kadaster ruimtelijk inzicht verschaffen in complexe registraties. Een complexe registratie is in dit geval een registratie waarbij meerdere subjecten betrokken zijn bij een perceel (eigenaren en/of pachters) en waarbij deze subjecten ieder een recht hebben dat is beperkt in de ruimte. Deze rechten liggen boven elkaar. De 3D-informatie is alleen nodig wanneer er sprake is van multifunctioneel ruimtegebruik. Bovendien is er al veel informatie bij het Kadaster aanwezig in twee dimensies (perceelsgrenzen, GBKN). Een uitbreiding van de bestaande 2D geo-DBMS om 3D geo-objecten te integreren zou een oplossing kunnen zijn voor de feitelijke behoefte aan 3D-informatie. Deze integratie bestaat uit het gecombineerd bevragen en visualiseren van de 2D- en 3D-gegevens.

[1)] Dit artikel is gebaseerd op de paper 'Incorporating 3D geo-objects into a 2D geo-DBMS' die gepresenteerd is tijdens het FIG XXII/ACSM-ASPRS-congres, 19 t/m 26 april 2002, Washington (VS).

3D-datamodellering

De belangrijkste DBMSs (Oracle, IBM DB2, Informix, Ingres) hebben geometrische datatypen en geometrische functies geïmplementeerd, die min of meer gelijk zijn aan de Simple Features Specifications voor SQL van het Open GIS Consortium (OGC) [2]. Deze implementaties bestaan uit een SQL-extensie voor de opslag, de bijhouding en het bevragen van Simple Features (punten, lijnen en polygonen). De OGC-specificaties zijn tot nog toe 2D, alhoewel er initiatieven worden ondernomen deze uit te breiden naar 3D. Ook de implementaties van ruimtelijke datatypen binnen DBMSs zijn voornamelijk 2D. De ondersteunde geometrische objecten in Oracle Spatial 9i [3],

dat in dit onderzoek wordt gebruikt, zijn punt, lijn en polygoon, inclusief cirkelbogen en combinaties van deze objecttypen (zie kader).

Zoals blijkt uit het voorbeeld in het kader kunnen z-waarden worden gebruikt om 3D-objecten (3D-lijnen, 3D-polygonen en 3D-punten) op te slaan. Echter 3D-geometrieprimitieven worden niet ondersteund in Oracle Spatial 9i. Daar-

Het object-relationele model in Oracle definieert een ruimtelijk object type sdo_geometry als:

```
CREATE TYPE sdo_geometry AS OBJECT (
SDO_GTYPE NUMBER,
SDO_SRID NUMBER,
SDO_POINT SDO_POINT_TYPE,
SDO_ELEM_INFO MDSYS.SDO_ELEM_INFO_ARRAY,
SDO_ORDINATES MDSYS.SDO_ORDINATE_ARRAY);
```

Waarbij:
sdo_gtype: het type geometrie (punt, lijn, polygoon, multipunt, multilijn, multipolygoon)
sdo_srid: het referentiesysteem dat gebruikt wordt voor de coördinaten
sdo_elem_info: een lijst met een aantal parameters die vertellen hoe de lijst met coördinaten geïnterpreteerd dient te worden. Een ruimtelijk object is samengesteld uit één of meerdere elementen. Voor elk element wordt aangegeven:

- bij welk coördinaat het element begint;
- het elementtype (punt, lijn,polygoon);
- een interpretatiecode (rechte lijn of een cirkelboog, rechthoek, enzovoort).

Een voorbeeld van het gebruik van het object-relationele model in Oracle is een rechthoek met (6,21) als linker onderhoek en (9,30) als rechter bovenhoek:

```
SDO_GEOMETRY Column = (
SDO_GTYPE = 2003
SDO_SRID = NULL
SDO_POINT = NULL
SDO_ELEM_INFO = (1,1003,3)
SDO_ORDINATES = (6,21,9,30))
```

In sdo_gtype=2003 geeft de eerste positie de dimensie aan (2D in dit geval), de laatste positie geeft het type weer (3 is een polygoon). In sdo_elem_info geeft de laatste 3 in 1,1003,3 aan dat dit een rechthoek is en daarom slechts twee punten nodig heeft voor de definitie (6,21 en 9,30). De 1003 (sdo_etype) geeft een buitengrens van een polygoon aan. De 1 (sdo_starting_offset) geeft aan dat dit element op positie 1 van de coördinatenlijst begint.

De volgende queries laten zien hoe een rechthoek (met 0,0 linksonder en 100,100 rechtsboven) wordt opgeslagen als een ruimtelijk datatype in Oracle (sdo_geometry type) in de 'geom2d'-tabel. Dezelfde rechthoek op een hoogte van 50 wordt opgeslagen in 'geom3d'. Eerst worden de tabellen aangemaakt, vervolgens gevuld, daarna worden de metadata opgeslagen en tenslotte worden de data geïndexeerd (d.m.v. een r-tree index die default is voor de ruimtelijke indexering). De metadata beschrijft de dimensies van de data en het bereik en de tolerantie van iedere dimensie.

```
/* creation of the tables */
create table geom2d (
  shape mdsys.sdo_geometry not null,
  TAG number(11) not null);
create table geom3d (
  shape mdsys.sdo_geometry not null,
  TAG number(11) not null);

/* inserting the data */
/* a 2D box */
insert into geom2d (shape,TAG) values (
  mdsys.SDO_GEOMETRY(2003, NULL, NULL,
  mdsys.SDO_ELEM_INFO_ARRAY(1, 1003, 3),
  mdsys.SDO_ORDINATE_ARRAY(0,0,100,100)),
1);
/* a 3D box */
insert into geom3d (shape,TAG) values (
  mdsys.SDO_GEOMETRY(3003, NULL, NULL,
  mdsys.SDO_ELEM_INFO_ARRAY(1, 1003, 3),
  mdsys.SDO_ORDINATE_ARRAY(0,0,50,100,100,50)),
2);

/* inserting metadata */
/* 2D table*/
insert into user_sdo_geom_metadata values
('GEOM2D', 'SHAPE', mdsys.sdo_dim_array(
mdsys.sdo_dim_element('X', 0, 500, 0.5),
mdsys.sdo_dim_element('Y', 0, 500, 0.5) ), NULL);
/*3D table*/
insert into user_sdo_geom_metadata values
('GEOM3D', 'SHAPE', mdsys.sdo_dim_array(
mdsys.sdo_dim_element('X', 0, 500, 0.5),
mdsys.sdo_dim_element('Y', 0, 500, 0.5),
mdsys.sdo_dim_element('Z', 0, 300, 0.5)
), NULL);

/* index creation */
/* geom2d*/
create index GEOM2D_I on GEOM2D(SHAPE)
  indextype is mdsys.spatial_index;
analyze table GEOM2D compute statistics;
/* geom3d*/
create index GEOM3D_I on GEOM3D(SHAPE)
  indextype is mdsys.spatial_index;
analyze table GEOM3D compute statistics;
```

Fig. 1.
De '3D'-rechten van het stationscomplex Den Haag Centraal. De verschillende kleuren in de foto geven aan welke 3D-objecten gedefinieerd zijn in het DBMS: het treinstation (geel): eigendom van NS Vastgoed, het spoor (blauw): eigendom van Railinfrabeheer, het bus- en tramstation (groen): eigendom van gemeente Den Haag en het kantorencomplex (rood): eigendom van Stichthage Trust bv.

om kunnen 3D-volumetrische datatypen niet worden gerepresenteerd en gemanipuleerd; 3D-volumetrische datatypen is dus toekomst voor DBMSen. Toch kan gebruik worden gemaakt van de voordelen van de huidige ondersteunde datatypen (3D-polygonen), door een 3D volume-object te representeren als een complex object: een polyhedron met referenties naar de vlakken waaruit het object bestaat. De vlakken worden vervolgens opgeslagen als 3D (vlakke) polygonen, elk als een apart record in de DBMS. Dit model is deels een topologisch model omdat het 3D geo-object wordt gedefinieerd door referenties naar de vlakken, maar de vlakken niet bestaan uit verwijzingen naar randen. Een andere mogelijkheid is het 3D geo-object te representeren als een multipolygoon (een polygoon dat bestaat uit meerdere polygonen), ook een ondersteund datatype in Oracle. In dat geval wordt een 3D-multipolygoon gedefinieerd dat bestaat uit meerdere 3D-polygonen, maar nu allemaal als één record in de DBMS.

Met deze oplossingen is het mogelijk om de 2D-topologische relatie te verkrijgen tussen (de projectie van) 3D geo-objecten en 2D geo-objecten door middel van de momenteel beschikbare ruimtelijke functies. Wanneer 3D-functionaliteiten eenmaal beschikbaar zijn, kunnen ook de 3D-ruimtelijke relaties worden verkregen. Een ander voordeel is dat de op deze manier opgeslagen 3D geo-objecten worden herkend door GIS- en CAD-applicaties die een databaseconnectie kunnen maken. Op deze manier kunnen de 2D- en 3D-data opgeslagen in het DBMS gecombineerd worden gevisualiseerd en bevraagd. Om dit te illustreren hebben we de rechten van het stationscomplex Den Haag Centraal (fig. 1) als 3D-ruimtelijke objecten opgeslagen in Oracle. De perceelsgrenzen en de contouren van gebouwen zijn ook opgeslagen als geometry datatypen (2D-lijnen) in het DBMS. Zoals uit fig. 2 blijkt is het mogelijk deze data te bevragen en te visualiseren in MicroStation Geographics, zowel in een 2D- als in een 3D-view.

Waar bevindt zich de 2D geo-data in de 3D-ruimte?

Bij het invoeren van 3D geo-objecten in de huidige 2D geo-DBMS komt de vraag op wat de verticale relatie is tussen de 2D geo-objecten (in dit geval percelen) en de 3D geo-objecten. Om te weten waar de 3D geo-objecten zich

Voorbeeld van een 3D-ruimtelijk object als multipolygoon

```
SQL> select * from geo-object3d;

Body_id
----------
SHAPE(SDO_GTYPE, SDO_SRID, SDO_POINT(X, Y, Z), SDO_ELEM_INFO, SDO_ORDINATES)
--------------------------------------------------------------------------------
1
SDO_GEOMETRY(3007,                                  --- 3007 indiceert een multipolygoon
NULL, NULL,
SDO_ELEM_INFO_ARRAY(1, 1003, 1, 37, 1003, 1, 52,    --- sdo_elem_info_array bestaat uit meerdere 1003,1
polygonen (buitenrand polygonen)
1003, 1, 67, 1003, 1, 82, 1003, 1, 97, 1003, 1, 112, 1003, 1, 127, 1003, 1, 142,
 1003, 1, 157, 1003, 1, 172, 1003, 1, 187, 1003, 1, 202, 1003, 1)
SDO_ORDINATE_ARRAY(82140.05, 455389.86, 0, 82124.4, 455378.31, 0, 82103.1, 455361.96, 0, 82036.91,
455311.16, 0, 82054.92, 455287.62, 0, 82063.07, 455293.84, 0, 82107.25, 455327.53, 0, 82151.73,
455361.4, 0, 82161.85, 455369.11, 0, 82159.77, 455371.79,0, 82143.72, 455392.57, 0, 82140.05, 455389.86,
0,                                                  -- hier eindigt eerste polygoon
82140.05, 455389.86, 0, 82140.05, 455389.86, 12, 82124.4,455378.31, 12, 82124.4, 455378.31, 0, 82140.05,
455 etc.....
```

wat de hoogte betreft bevinden, moet het horizontale nulniveau worden gedefinieerd waarop de 2D geo-objecten zich bevinden. Er zijn twee mogelijkheden om de z-coördinaten die de geo-objecten in 3D definiëren, te representeren.

Een absolute z-coördinaat gedefinieerd t.o.v. NAP

Wanneer de z-coördinaten van 3D geo-objecten worden opgeslagen in NAP, moet ook de hoogte van de percelen aan het maaiveld bekend zijn in NAP om de geometrische en topologische relatie te kunnen vinden tussen de percelen en de 3D geo-objecten, zoals: 'ligt boven', 'ligt onder' of 'snijdt'. De inwinning en de invoer van deze toegevoegde informatie gerelateerd aan de 2D-percelen, zullen een aanzienlijke tijd kosten. Bovendien neemt de complexiteit van de 2D-data toe, omdat 2D-percelen gedefinieerd moeten worden in de 3D-ruimte. Dit kan niet worden gedaan door simpelweg één z-coördinaat toe te voegen per perceel, omdat de meeste percelen te veel variantie in hoogte bevatten.

Een relatieve z-coördinaat gedefinieerd t.o.v. het maaiveld

Wanneer z-coördinaten van de 3D geo-objecten worden opgeslagen ten opzichte van het maaiveld, hoeft de huidige geo-DBMS niet te worden uitgebreid met informatie over de hoogte voor de reeds beschikbare 2D geo-data. Dit bespaart tijd, maar wat belangrijker is, ook complexiteit. De z-coördinaten van de 3D geo-objecten die bekend zijn in NAP, moeten worden geconverteerd naar relatieve z-coördinaten. In dit geval hoeft alleen de directe omgeving van de 3D-situatie te worden geëxploreerd in plaats van alle percelen in het DBMS te positioneren in de 3D-ruimte. Nadeel is wel dat dataconsistentie moeilijk kan zijn. Wanneer bijvoorbeeld het maaiveld door vergravingen verandert, levert dit niet automatische een update op bij het Kadaster.

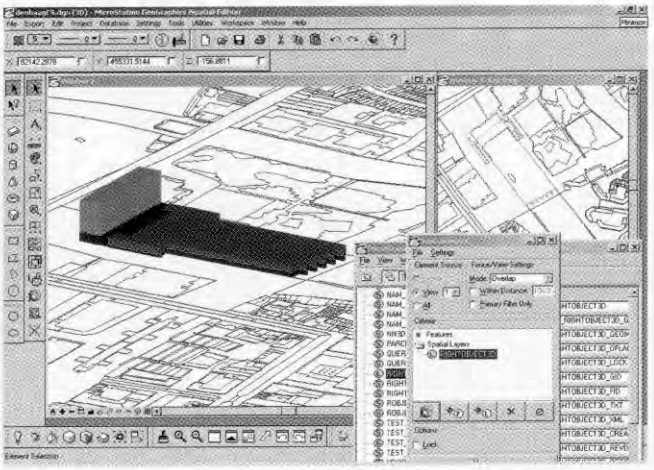

Fig. 2. Het gebruik van CAD (MicroStation) om 3D en 2D geo-objecten opgeslagen in het DBMS te visualiseren (donkerblauw: NS Vastgoed; paars: NS Railinfrabeheer; groen: gemeente Den Haag; lichtblauw: Stichthage Trust bv.).

Een relevante vraag is ook waar het horizontale nulniveau (maaiveld) zich precies bevindt. Vooral in 3D-situaties kan het moeilijk zijn het maaiveld eenduidig te definiëren, bijvoorbeeld gebouwen met een complexe architectuur, die half onder de grond zijn gebouwd. Binnen dit onderzoek is een experiment uitgevoerd, waarbij de coördinaten die de kadastrale percelen beschrijven een z-coördinaat toegekend hebben gekregen, dat is gegenereerd uit het AHN [4]. Hiervoor is een TIN gebouwd op het basisbestand van het AHN. Daarnaast hadden we de beschikking over de x,y,z-informatie (ook in NAP) van een NAM-pijpleiding. Door deze informatie kon de NAM-leiding worden gelokaliseerd ten opzichte van de percelen (fig. 3). Daarnaast hebben we de maaiveldhoogte verkregen uit het AHN op de locatie van de NAM-leiding, zodat ook de ligging van deze leiding ten opzichte van het maaiveld bepaald kon worden (fig. 4).

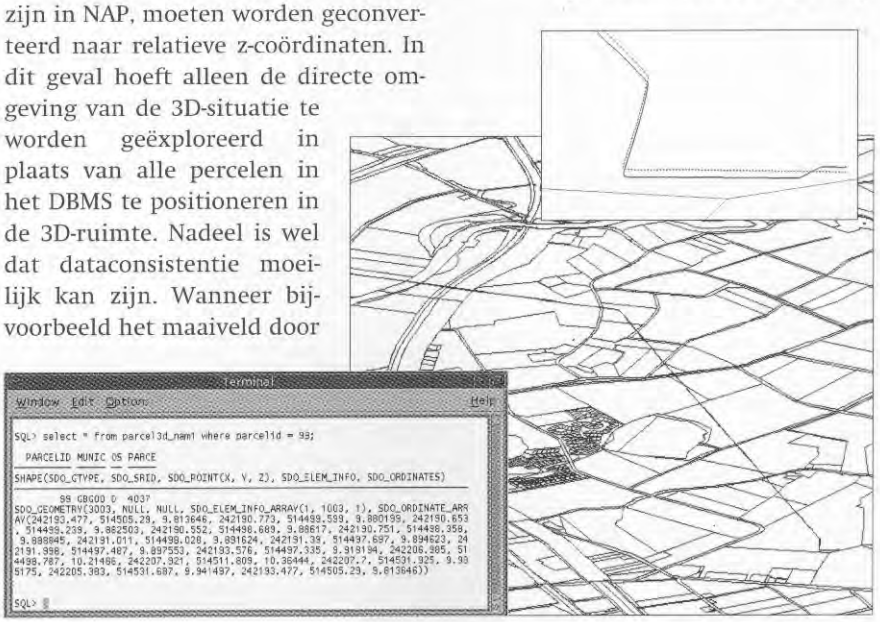

Fig. 3. Percelen opgeslagen in het DBMS en gedefinieerd met x,y,z-coördinaten (in NAP), gecombineerd met een NAM-pijpleiding (ook bekend in NAP). De stippellijn is de maaiveldhoogte op de locatie van de NAM-leiding. De dieptelligging van de leiding is moeilijk te zien omdat de perceelsgrenzen in 3D zijn weergegeven en niet de vlakken van de percelen.

Een gecombineerde bevraging van 2D en 3D geo-data

Het beheren van zowel impliciete als expliciete relaties tussen de 3D geo-objecten en de 2D-percelen maakt de integratie van 3D geo-objecten in de 2D geo-DBMS compleet. Door deze relaties is het mogelijk de data te bevragen in 2D- en 3D-mode, en ook administratief. Omdat de momenteel beschikbare ruimtelijke functies slechts in 2D werken, kan alleen in 2D de ruimtelijke relatie tussen de 2D-percelen en 3D geo-objecten worden bevraagd (dus de relatie tussen de 2D-percelen en de projectie van de 3D geo-objecten). De ruimtelijke functies in Oracle werken wel op de 3D geo-objecten zoals deze geïmplementeerd zijn, maar bij de berekeningen (bijvoorbeeld oppervlakte of afstandsberekening) wordt de z-waarde niet meegenomen; 2D-topologische relaties zijn geïmplementeerd door middel van de sdo_relate operator (of de sdo_geom.relate functie). Wanneer

deze operator wordt gebruikt op 3D-punten, 3D-lijnen of 3D-(multi)polygonen, worden deze 3D-objecten eerst geprojecteerd op het oppervlak.

Het huidige 2D-kadastrale datamodel, waarin de percelen zijn gedefinieerd, is gebaseerd op een topologische structuur waarbij de percelen topologisch worden opgeslagen (door middel van de kettingstructuur met een verwijzing naar een eerste lijnstuk vanuit het perceelsvlak). De perceelsgrenzen zijn geometrisch opgeslagen [5]. Het is mogelijk om de percelen te selecteren, die overlappen met de projectie van een 3D geo-object (zoals de 3D geo-objecten rond Den Haag Centraal) door middel van een sql-query. Hiervoor moeten de percelen eerst 'gerealiseerd' of 'gematerialiseerd' worden. Dat betekent dat de geometrie van een perceel wordt gegenereerd op basis van de topologische relaties en de onderliggende perceelsgrenzen. Hiervoor hebben we een functie geschreven in pl/sql [6]. De onderstaande query laat zien dat het mogelijk is percelen te selecteren, die overlappen met de (projectie van) 3D geo-objecten. De functie 'return_polygon' realiseert de geometrie van de percelen op basis van de tabel waarin de perceelsgrenzen staan opgeslagen.

```
/* perceel selectie op basis van overlap met de projectie
van een 3D object */

select distinct municip, section, parcel_number
from lki_parcel p, geom3d g3d
where sdo_geom.relate(return_polygon(p.object_id),
                      'anyinteract',g3d.shape,.1)='TRUE';
```

Wanneer je eenmaal hebt uitgevonden welke percelen doorsneden worden door een 3D geo-object, kun je ook de juridische relaties nader bekijken door middel van administratieve queries op de tabellen die informatie bevatten over de percelen, de 3D geo-objecten en hun eigenaren.

Extensie naar een 3D-datatype

Nu volgt er een voorstel om de huidige datatypen van Oracle uit te breiden met ondersteuning voor echte 3D geo-objecten. Hoewel we hierbij Oracle als voorbeeld gebruiken om de uitbreiding uit te leggen en te implementeren, zijn de ideeën achter dit voorstel generiek en gelden deze ook voor andere DBMSs.

De ondersteuning voor 3D geo-objecten houdt de geometrische definitie van een 3D-volume primitieve in. Dit is een toevoeging op de 0D-, 1D- en 2D-primitieven zoals punt, lijn en polygoon, die ook beschikbaar moeten zijn in de 3D-ruimte. Dat betekent dat deze primitieven gedefinieerd zijn door middel van x-,y- en z-waarden, dat deze primitieven door het DBMS gevalideerd kunnen worden en dat de beschikbare functies (oppervlakte-, afstandsberekening, enzovoort) ook in de 3D-ruimte werken en niet op het geprojecteerde vlak z=0. Bij de introductie van een 3D-volumeprimitieve zijn er verschillende mogelijkheden. In de CAD-wereld waar 3D-objecten meer gangbaar zijn [7], zijn er enkele benaderingen voor de 3D-modellering: 'boundary' (grens) representatie, CSG (constructive solid geometry), regelmatige 3D-grids, enzovoort. In de GIS-wereld is het geometrische model van OpenGIS, dat eerder in dit artikel is beschreven, een breed geaccepteerd model. Oracle heeft een soortgelijke implementatie hiervan voltooid tot en met de 2D-primitieve en de bijbehorende functies. De 3D-uitbreiding moet passen in het bestaande geometrische model. Van relatief simpele tot complexe uitbreidingen kan gedacht worden aan de volgende oplossingen:

- introductie van de tetrahedron primitieve (viervlak, bestaande uit driehoeken): dit is de meest simpele 3D-volume primitieve en de implementatie van functies (volume, oppervlak, intersectie, afstand, buffer) zijn relatief ook vrij eenvoudig. Voor het modelleren van een object

Fig. 4. NAM-leiding gecombineerd met AHN geeft als resultaat waarden op het maaiveld en waarden ten opzichte van het maaiveld op de locatie van de NAM-leiding. Merk op dat het eerste gedeelte zich boven het maaiveld bevindt: de leiding komt op het NAM-terrein bovengronds.

x	y	z (NAP)	mv (NAP)	z_tov_maaiveld
242850.36	512938.67	10.44	9.177488	1.262512
242850.25	512938.80	10.34	9.17832	1.16168
242849.52	512939.37	10.35	9.181968	1.168032
242847.22	512941.22	10.34	9.17208	1.16792
242847.21	512941.25	10.38	9.172072	1.207928
242847.17	512941.25	10.38	9.171944	1.208056
242845.24	512942.82	10.38	9.165616	1.214384
242844.80	512943.11	10.23	9.163164	1.066836
242843.01	512944.55	8.89	9.165188	-0.275188
242841.22	512945.95	7.58	9.170208	-1.590208
242840.92	512946.21	7.42	9.170436	-1.750436
242840.47	512946.54	7.33	9.17049	-1.84049
242839.11	512947.64	7.32	9.170586	-1.850586
242835.29	512950.64	7.23	9.197065	-1.967065
242821.29	512962.05	7.17	9.140288	-1.970288
242820.76	512962.44	7.16	9.119136	-1.959136
242811.67	512969.86	6.93	8.96213	-2.03213
242807.54	512973.22	6.89	8.942776	-2.052776
242806.95	512973.71	6.87	8.941536	-2.071536
242806.38	512974.91	6.87	8.949648	-2.079648
242806.37	512974.91	6.88	8.949504	-2.069504
242806.71	512976.23	6.88	8.972352	-2.092352
242808.41	512978.28	6.92	8.995645	-2.075645
242809.12	512979.15	6.94	8.989553	-2.049553
242813.98	512985.07	7.01	8.977824	-1.967824
242820.15	512992.63	7.06	8.989792	-1.929792
242826.80	513000.78	7.07	9.005609	-1.935609
242834.26	513010.03	7.09	9.021051	-1.931051
242841.85	513019.50	7.04	8.98325	-1.94325
242849.28	513028.84	7.12	9.031892	-1.911892
242856.49	513038.14	7.16	8.95139	-1.79139
242863.71	513047.60	7.12	8.973624	-1.853624
242864.57	513048.72	7.12	8.932208	-1.812208
242865.71	513050.42	7.11	8.898592	-1.788592
242865.71	513050.43	7.10	8.898578	-1.798578
242866.62	513052.32	7.10	8.903212	-1.803212
etc.....				

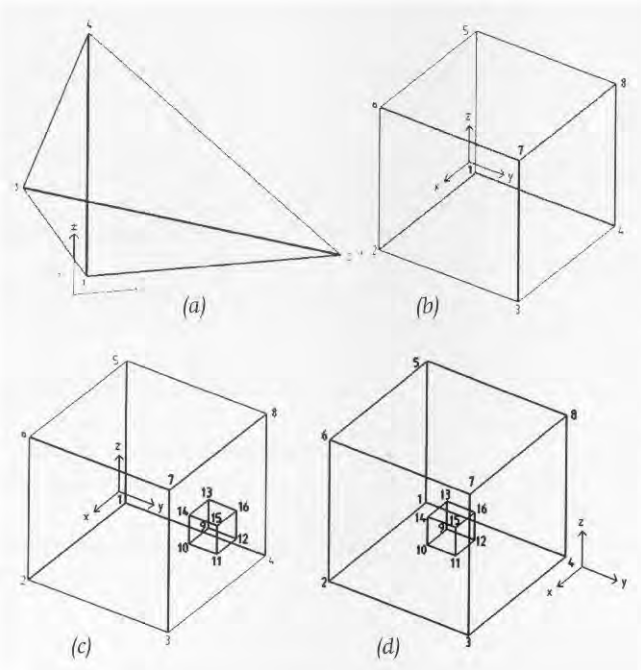

Fig. 5.
Een aantal voorbeelden van een 3D-object bestaande uit een polyhedron.

kan echter wel een groot aantal tetrahedrons nodig zijn;

- introductie van de polyhedron primitieve (3D-object bestaande uit platte vlakken): dit is de 3D-equivalent van de 2D-polygoon. De grenzen zijn gedefinieerd door platte vlakken (polygonen in 3D). Omdat de grenzen ook 'gaten' kunnen bevatten, is deze oplossing al vrij complex. Ook zou er sprake kunnen zijn van één of meerdere binnengrenzen voor gaten binnenin het 3D-object zelf. De voordelen van deze oplossing is dat er slechts één polyhedron nodig is om een wereldobject te modelleren. Niet-vlakke polygonen moeten worden benaderd door een aantal vlakke polygonen;
- introductie van een polyhedron, een cylinder en een bol, en de combinatie van deze drie. Dit is de 3D-equivalent van het geometriemodel in 2D die ook cirkelbogen ondersteunt. Het voordeel van dit model is dat grootschalige modelleringen veel van dit soort geo-objecten kennen.

Door de modelleervoordelen en de analogie met het 2D-geometriemodel hebben we eerst de extensie met polyhedrons, bollen en cilinders bestudeerd in analogie met 2D-punten, lijnen (rechte en cirkelbogen) en polygonen. Het is echter moeilijk voor te stellen dat een gebruiker een 3D-primitieve beschrijft, waarvan de grens bestaat uit een combinatie van platte, cilindrische en bolvormige delen. De grens dient natuurlijk gesloten te zijn. De implementatie van bollen en cilinders zal niet het moeilijkste zijn; het gebruik van cilindrische en bolvormige delen (in analogie met een 2D-lijn als deel van een cirkel) daarentegen is gecompliceerd. Twee delen moeten bijvoorbeeld altijd aansluiten op een gemeenschappelijke lijnstuk. Daarom stellen we voor om het geometrische datamodel van Oracle uit te breiden met alleen de polyhedron-primitieve. In de toekomst zou gekeken kunnen worden naar meer complexe geometrie, natuurlijk aansluitend op de OpenGIS-ontwikkelingen op dit gebied.

Om de polyhedron-oplossing nader te bestuderen hebben we gekeken naar vier voorbeeld-objecten (fig. 5):

1. tetrahedron, het meest simpele 3D-object met een volume, bestaande uit vier vlakken: driehoeken in 3D (fig. 5a);
2. een blok: een volume met zes vlakken: rechthoeken in 3D (fig. 5b);
3. een blok zoals (2), maar dan met een gat in één van de grensvlakken. Het totale object heeft elf vlakken, die alle deel uit maken van de buitengrens (fig. 5c);
4. een blok zoals (2), maar nu met een gat binnenin het object: in totaal heeft het object twaalf vlakken: zes vlakken maken deel uit van de buitengrens en zes vlakken maken deel uit van de binnengrens (fig. 5d).

Merk op dat een plat vlak in 3D moeilijk te definiëren is omdat, wanneer een vlak gedefinieerd wordt door meer dan drie punten, het niet meer vanzelfsprekend is dat deze punten in hetzelfde vlak liggen.

Een polyhedron wordt gedefinieerd door de grenselementen (vlakken). Deze kunnen buitenvlakken zijn (fig. 5a), maar ook binnenvlakken (fig. 5d, het gat binnenin het blok). Verder kan een buitenvlak ook weer binnenringen bevatten (fig. 5c, rechterzijde). Een binnenvlak kan natuurlijk ook een binnenring bevatten (niet als voorbeeld aanwezig). De polyhedron wordt net als de lijnen en de polygonen gedefinieerd door een coördinatenlijst. Deze is impliciet genummerd van 1 tot N. Deze lijst wordt aangevuld met de definitie van de vlakken, die bestaat uit het noemen van alle puntnummers die voorkomen in een vlak. Deze codering zorgt voor een interne topologie van de 3D-primitieve. De interpretatiecode in de sdo_elem_info geeft aan hoe de coördinatenlijst geïnterpreteerd dient te worden. In het kader staat beschreven hoe dit voorgestelde datamodel er uitziet, toegepast op de objecten in fig. 5. In de voorbeelden hebben we gemakshalve de oriëntatie van de vlakken buiten beschouwing gelaten, die beschreven kan worden door de ordening van de punten in een vlak. De buitenvlakken zouden een normaal moeten hebben die naar buiten wijst (coördinaten met de klok mee) en de binnenringen een normaal die naar binnen wijst.

Conclusie

Het Kadaster is op zoek naar een manier om meer inzicht te krijgen in complexe situaties. Dit betekent dat het mogelijk moet zijn om de ruimtelijke component (2D en 3D) van ge-

Codering van een polyhedron

De polyhedron wordt gedefinieerd met sdo_gtype d008 (d=3 voor 3D of d=4 voor 4D). (In analogie met het 2D-model wordt de multigeometrie variant gecodeerd met sdo_gtype=d009. Deze wordt hier verder buiten beschouwing gelaten).

De meer gedetailleerde codering van het polyhedron wordt gerealiseerd door de elementtypen sdo_etype. Het voorstel is om sdo_etype=xxx6 te gebruiken voor een grenselement, xxx=100 kan daarbij gebruikt worden voor een buitenring van een buitenvlak en xxx=200 voor de buitenring van een binnenvlak (voorbeeld 5d).

De binnenringen van buitenvlakken (voorbeeld 5c) coderen we met etype=1106. Binnenringen van binnenvlakken worden op dezelfde wijze gecodeerd met etype=2106.

De sdo_interpretationcode vertelt nu hoe de coördinaten in de coördinatenlijst geïnterpreteerd moeten worden. Voor de polyhedron stellen we de volgende sdo_interpretation codes voor: 3 voor 3D-blokken (vergelijk met de rechthoek in 2D), en 1 voor polyhedrons met platte vlakken (vergelijk met de polygoon met rechte lijnsegmenten in 2D). De voorgestelde nummers voor gtype, etype en de interpretatiecode zien er misschien wat vreemd uit, maar deze zijn in analogie met het geometriemodel in 2D, waarbij we de bolvormige en cilindrische delen voor dit moment overslaan.

Het vullen van de coördinatenlijst (sdo_ordinate_array) is niet triviaal. Om te voorkomen dat coördinaten van twee aangrenzende vlakken dubbel worden genoemd, begint de lijst eerst met het noemen van alle voorkomende coördinaten, die impliciet genummerd zijn van 1 tot N. Het startpunt van elk vlak is gegeven in de sdo_elem_info_array. Een vlak wordt gespecificeerd door het noemen van alle puntnummers. Deze codering zorgt voor een interne topologie van de 3D-primitieve.

Nu het meest eenvoudige voorbeeld van fig. 5a: de tetrahedron met de volgende vier coördinaten: 1=(0,0,0), 2=(1,0,0), 3=(0,1,0) en 4=(0,0,1):

```
insert into geom3d (shape, TAG) values (
  mdsys.SDO_GEOMETRY(3008, NULL, NULL,
                    -- polyhedron, zonder ref. point en srid
    mdsys.SDO_ELEM_INFO_ARRAY(13,1006,1
                    -- eerste platte vlak begint op offset 13,
    -- eerste 12 posities zijn gebruikt voor de coördinaten
        16,1006,1, 19,1006,1, 22,1006,1),
                    -- de ander vlakken op 16, 19 en 22
    mdsys.SDO_ORDINATE_ARRAY(0,0,0,
                    -- coordinaat triplet van punt 1
      1,0,0, 0,1,0, 0,0,1,          -- en van punt 2, 3 en 4
      1,2,3, 1,2,4, 1,3,4, 2,3,4)),
                -- de vlakken referenties naar de punten
  1);                               -- TAG van voorbeeld 1
```

De coördinaten van het blok in fig. 5b zijn: 1=(0,0,0), 2=(5,0,0), 3=(5,5,0), 4=(0,5,0), 5=(0,0,5), 6=(5,0,5), 7=(5,5,5) en 8=(0,5,5). Het coderen in Oracle ziet er als volgt uit:

```
insert into geom3d (shape, TAG) values (
  mdsys.SDO_GEOMETRY(3008, NULL, NULL,
    mdsys.SDO_ELEM_INFO_ARRAY(25,1006,1
                                -- eerste vlak op offset 25
        29,1006,1, 33,1006,1, 37,1006,1, 41,1006,1,
                                                45,1006,1),
    mdsys.SDO_ORDINATE_ARRAY(0,0,0,
                    -- coordinaten triplet van punt 1
      5,0,0, 5,5,0, 0,5,0, 0,0,5, 5,0,5, 5,5,5, 0,5,5,
      1,2,3,4, 5,6,7,8, 4,3,7,8,
                                -- onder, boven, rechter vlak
      1,2,6,5, 1,4,8,5, 2,3,7,6)),
                                -- linker, achter en voor vlak
```

vestigde rechten (en beperkingen en belemmeringen) die kenbaar zijn uit de Openbare Registers en de kadastrale registratie te visualiseren. In dit artikel hebben we beschreven hoe beschikbare geo-informatie in 2D gecombineerd kan worden met 3D geo-informatie om aan deze informatiebehoefte te voldoen. We hebben ook een uitbreiding van het huidige geometrische datamodel in DBMSs voorgesteld om de ondersteuning van 3D volume-objecten mogelijk te maken. Volledig 3D-support in DBMSs zal echter nog wel even op zich laten wachten. In de tussentijd zijn we bezig om binnen de huidige technieken de mogelijkheid te implementeren om 3D geo-objecten te beheren in het DBMS en deze te combineren met de bestaande 2D-data (de percelen). Op deze manier kunnen verschillende vragen worden beantwoord en gevisualiseerd: 'tot welke hoogte en diepte geldt het opstalrecht op dit perceel', 'wordt de ondergrond van dit perceel ook doorsneden door de pijpleiding', 'welke percelen zijn belast met een beperking in het recht van eigendom ten gevolge van een tunnel' en 'hoe is deze beperking gedefinieerd in 3D' worden beantwoord en gevisualiseerd. Dit alles moet op termijn voor een meer duurzame oplossing zorgen in een wereld met een toenemende druk op het ruimtegebruik. De registratie wordt eenvoudiger en eenduidiger, maar ook meer toegankelijk en inzichtelijk omdat deze beter aansluit bij de werkelijkheid. ■

Literatuur

[1] Stoter, J. E., M. A. Salzmann, P. J. M. van Oosterom en P. van der Molen, *Op naar een 3D-Kadaster? Registratie van multi-functioneel ruimtegebruik.* Geodesia 2002 no. 3, p. 88-93.

[2] OGC, *OpenGIS Simple Features Specification for SQL. Revision 1.1.* OpenGIS Project Document 99-049.

[3] Oracle, *Oracle Spatial User's Guide and Reference Release 9.0.1 Part Number A88805-01.* Juni 2001.

```
2);                                    -- TAG van voorbeeld 2
```

Het volgende voorbeeld (fig. 5c) bevat een buitenvlak met een binnenring. Per definitie hoort de binnenring tot het vlak dat voorafgaat. Dit vlak moet ook in hetzelfde draagvlak gedefinieerd zijn.
De coördinaten van de eerste acht punten zijn gelijk als in voorbeeld 5b en de daaropvolgende coördinaten zijn: 9=(2,4,2), 10=(3,4,2), 11=(3,5,2), 12=(2,5,2), 13=(2,4,3), 14=3,4,3), 15=(3,5,3), en 16=(2,5,3). De binnenring begint op offset 61 (etype=1106). Deze polyhedron heeft alleen buitenvlakken (etype 1006 en 1106) en geen binnenvlakken als begrenzing (etype 2006, mogelijk gevolgd door etype 2106). De code ziet er als volgt uit:

```
insert into geom3d (shape, TAG) values (
 mdsys.SDO_GEOMETRY(3008, NULL, NULL,
                    -- polyhedron, geen ref. point, geen srid
  mdsys.SDO_ELEM_INFO_ARRAY(49,1006,1
                          -- eerste platte vlak op offset 49
   53,1006,1, 57,1006,1, 61,1106,1,
             -- vlak beginnend op 57 bevat binnenring 61
 65,1006,1, 69,1006,1, 73,1006,1,
       77,1006,1, 81,1006,1, 85,1006,1, 89,1006,1,
                                                93,1006,1),
  mdsys.SDO_ORDINATE_ARRAY(0,0,0,
                          -- coordinaten triplet van punt 1
   5,0,0, 5,5,0, 0,5,0, 0,0,5, 5,0,5, 5,5,5, 0,5,5,
   2,4,2, 3,4,2, 3,5,2, 2,5,2, 2,4,3, 3,4,3, 3,5,3, 2,5,3,
   1,2,3,4, 5,6,7,8, 4,3,7,8,
                           -- onder, boven en rechter vlak
   11,12,16,15          -- binnenring van rechtervlak
   1,2,6,5, 1,4,8,5, 2,3,7,6,
                        -- linker, achter en voor vlak
   13,14,15,16, 9,10,11,12,          -- gat: boven, onder
   9,10,14,13, 9,12,16,13, 10,11,15,14)),
                                  -- gat: links, achter, voor
 3);                              -- TAG van voorbeeld 3
```

Het laatste voorbeeld laat de polyhedron zien met een gat binnenin (fig. 5d). De eerste acht punten zijn weer gelijk aan de eerdere voorbeelden, en de coördinaten van het gat zijn: 9=(2,2,2), 10=(3,2,2), 11=(3,3,2), 12=(2,3,2), 13=(2,2,3), 14=(3,2,3), 15=(3,3,3), en 16=(2,3,3). De buitengrens is weer beschreven met zes platte vlakken (etype 1006). De binnenring is ook beschreven met zes platte vlakken (etype 2006).

```
insert into geom3d (shape, TAG) values (
 mdsys.SDO_GEOMETRY(3008, NULL, NULL,
                    -- polyhedron, geen ref. point, geen srid
  mdsys.SDO_ELEM_INFO_ARRAY(49,1006,1
                          -- eerste platte vlak op offset 49
   53,1006,1, 57,1006,1, 61,1006,1, 65,1006,1,
   69,1006,1, 73,2006,1, 77,2006,1, 81,2006,1,
   85,2006,1, 89,2006,1, 93,2006,1),
  mdsys.SDO_ORDINATE_ARRAY(0,0,0,
                          -- coordinaten triplet van punt 1
   5,0,0, 5,5,0, 0,5,0, 0,0,5, 5,0,5, 5,5,5, 0,5,5,
   2,2,2, 3,2,2, 3,3,2, 2,3,2, 2,2,3, 3,2,3, 3,3,3, 2,3,3,
   1,2,3,4, 5,6,7,8, 4,3,7,8,
                           -- onder, boven rechter vlak
   1,2,6,5, 1,4,8,5, 2,3,7,8,
                        -- linker, achter en voor vlak
   9,10,11,12, 13,14,15,16, 11,12,16,15,
                           -- gat: onder, boven, rechts
   9,10,14,13, 9,12,16,13, 10,11,15,14)),
                                  -- links, achter en voor
 4);                              -- TAG van voorbeeld 4
```


[4] Heerd, R.M. van et al., *Productspecificatie AHN 2000.* Rijkswaterstaat, Meetkundige Dienst, rapportnummer: MDTGM 2000.13, 2000.

[5] Oosterom, P.J.M. van, en C.H.J. Lemmen, *Spatial data management on a very large cadastral database.* Computers, Environments and Urban Systems (CEUS), 2001 volume 25, number 4-5, p. 509-528.

[6] Oosterom. P.J.M. van, J.E. Stoter, S. Zlatanova, W. Quak, *The balance between Geometry and Topology.* Spatial Data Handling 2002 Symposium, Ottowa, Canada, 9-12 juli 2002.

[7] Mortenson, M.E., *Geometric Modeling,* John Wiley, New York 1985.

Summary

Databases combining 2D and 3D geodata

While geodata is extensively available in 2D only, increasing multiple landuse requires registration of the vertical component of objects. The authors describe their research that aims at including 3D information in a 2D DBMS. The concept is demonstrated with a prototype in Oracle Spatial 9i.

KEYWORDS
GIS-technology, data structures, system development

TREFWOORDEN
GIS-technologie, gegevensstructuren, systeembeschrijving