Consistentie moet absoluut gewaarborgd zijn

Echte Relationele Databases

Ivan Pellegrin

"Is dit echt pure wol?" vroeg grootmoeder met een twijfelende uitdrukking op haar gezicht, terwijl ze de stof tussen haar vingers wreef. Ik heb grootmoeder deze truc al vele keren zien uithalen. Het werkte altijd. "Natuurlijk, mevrouw", antwoordde het jonge meisje gretig: "100 procent pure wol!" Die is vast nieuw hier, dacht ik. "Ach wat jammer nou", sprak grootmoeder, op een toon die deed denken aan de geduchte Miss Marple die een schier onoplosbaar moordmysterie oploste. "Ik was op zoek naar iets half-om-half." Grootmoeder hield niet van wol, om redenen die altijd gehuld bleven in geheimzinnigheid.

Om net zulke onduidelijke redenen wendt een groot aantal dataprofessionals het hoofd in afgrijzen af, als ze iemand 'Echte Relationele Databases' horen zeggen. Velen vermijden een gesprek over Echte Relationele Databases, net zoals veel acteurs de naam van Het Schotse Toneelstuk niet hardop durven zeggen. Sommigen leven in het zalige geloof dat op SQL gebaseerde producten echt relationeel zijn. Aan de andere kant zijn de software vendors niet bepaald behulpzaam geweest in deze kwestie, door hun op SQL gebaseerde producten voortdurend van het etiket 'relationeel' te voorzien. Anderen denken dat het onderwerp puur academisch is. "Het raakt me niet", hoor ik ze zeggen, "Professoren en studenten hebben de tijd om over dat soort dingen na te denken. Ik kan er trouwens weinig aan veranderen. Ik laat het zijn werk doen met datgene dat ik nu tot mijn beschikking heb."
In dit artikel zal ik dit standpunt aanvechten en aantonen dat het u wel degelijk raakt. Ook bestrijd ik het standpunt dat we er niets aan zouden kunnen doen. Maar allereerst geef ik een kort overzicht van de geschiedenis van het Relationele Model en een inleiding over enkele van haar fundamentele concepten.

Geschiedenis van het Relationele Model

Deze paragraaf beoogt een (erg persoonlijk) beeld te geven van de geschiedenis van het Relationele Model om enkele belangrijke gebeurtenissen wat perpectief te geven. Ik heb niet de pretentie om volledig te zijn en het vat mijn eigen studies samen.
In 1969 publiceerde Edgar Codd dat wat terecht beschouwd wordt als de originele beschrijving van het Relationele Model met de titel 'Derivability, Redundancy, Consistency of Relations Stored in Large Data Banks'. Ik wil graag de aandacht vestigen op de zorgvuldige woordkeuze in de titel: afleidbaarheid, overtolligheid en consistentie zijn nog net zo significant – en ongrijpbaar – als bijna 40 jaar geleden.
In 1970 publiceerde Codd een bewerkte versie van de eerdere publicatie onder de titel 'A Relational Model of Data for Large Shared Data Banks'. Dit academische artikel verwierf grotere bekendheid dan de eerste publicatie en wordt gezien als de oorsprong van het Relationele Model.
In augustus 1998 publiceerden Chris Date en Hugh Darwen een preview van The Third Manifesto. Later dat jaar publiceerden zij de eerste editie van The Third Manifesto. De volledige titel luidt 'Foundation for Object/Relational Databases: the Third manifesto – A Detailed Study of the Impact of Objects and Type Theory on the Relational Model of Data Including a Comprehensive Proposal for Type Inheritance'. Hoewel de titel te lijden had van de objectgeoriënteerde hype die de database-wereld in de tweede helft van de jaren negentig teisterde, zette de inhoud van het boek de belangrijkheid van het Relationele Model opnieuw in het middelpunt. De auteurs verwoorden dit zelf als "The Third Manifesto is a formal proposal (...) for a solid foundation for data and database management systems" [1].

In 2000 publiceerden Date en Darwen de tweede editie met een wat gelukkiger titel die kort genoeg was om in één adem uit te spreken: 'Foundation for Future Database Systems: The Third Manifesto'. De auteurs raakten echter ontevreden over deze titel, omdat de toekomst zich nooit zal voordoen; in 2006 publiceerden Date en Darwen een derde herziene versie met de titel 'Databases, Types and the Relational Model: the Third Manifesto'. Toekomstige edities van dit boek zullen misschien een wat meer merkgerichte titel krijgen zoals bijvoorbeeld 'The Third Manifesto', met alle verwijzingen naar databases, types, relationele modellen enzovoort alleen in de metadata van het boek.
Hoewel Date's en Darwen's Third Manifesto niets van de politieke lading heeft waarmee de term 'manifesto' meestal geassocieerd wordt, bevat hun werk aspecten die net zo radicaal en revolutionair zijn. De toepassing van hun principes kan het DBMS-landschap zoals we dat nu kennen, fundamenteel veranderen. De lezer moet zich niet uit het veld laten slaan door de

omvang van het boek, maar het bijwonen van een seminar over dit onderwerp zou een snelle en grondige kennis van de inhoud aanzienlijk makkelijker maken.

Inleiding tot het Relationele Model

Een volledige beschrijving van het Relationele Model valt ver buiten het kader van dit artikel. Desondanks wil ik de aandacht vestigen op een top-12 van fundamentele concepten.

1. Het relationele datamodel maakt het mogelijk een logische en consistente representatie van informatie te creëren.
2. Een type is een benoemde, eindige verzameling waarden. INTEGER, bijvoorbeeld, is de verzameling van alle integers die door een gegeven computersysteem kunnen worden voorgesteld, CHAR is de verzameling van alle karakterstrings die door een gegeven computersysteem kunnen worden voorgesteld, S# is de verzameling van alle supplier ID's, enzovoort. Een speciaal geval is het lege type dat bestaat uit een lege verzameling waarden.
3. Een verzameling is een ongeordende collectie van items. Dat betekent dat (a, b) en (b, a) twee representaties zijn van dezelfde verzameling.
4. Een attribuut is een verzameling van attribuutnaam en typenaam, bijvoorbeeld:
   ```
   (S_ID S#)
   ```
5. Een 'heading' is een verzameling attributen. In de wetenschap dat een verzameling ongeordend is, zijn dit (niet de enige) twee representaties van dezelfde heading:
   ```
   {S_ID S#, S_NAME CHAR, S_STATUS INTEGER}
   {S_ID S#, S_STATUS INTEGER, S_NAME CHAR}
   ```
6. Een 'tuple' is een verzameling attribuutwaarden. Dit is een voorbeeld van een mogelijke representatie van een tuple:
   ```
   TUPLE {S_ID S#('S1'), S_NAME 'Joe Blog',
                                        S_STATUS 10}
   ```
7. Een relatie bestaat uit een 'heading' en een 'body'. Dit is een mogelijke representatie van een relatie-heading:
   ```
   {S_ID S#, S_NAME CHAR, S_STATUS INTEGER}
   ```
8. De 'body' van een n-aire relatie is een verzameling n-tuples. Een relatie-body zou er zo uit kunnen zien:
   ```
   RELATION {
   TUPLE {S_ID S#('S1'), S_NAME 'Titius',
                                        S_STATUS 10},
   TUPLE {S_ID S#('S2'), S_NAME 'Caius',
                                        S_STATUS 20},
   TUPLE {S_ID S#('S3'), S_NAME 'Sempronius',
                                      S_STATUS 10} }
   ```
9. In het Relationele Model worden alle data voorgesteld als relaties.
10. Het redeneren over alle data wordt gedaan in logica met twee predikaatwaarden. Dat betekent dat elke bewering enkel en alleen maar twee mogelijke uitkomsten kan hebben: WAAR of NIET WAAR. Er bestaat in het Relationele Model geen NULL-waarde, vaak gebruikt als onbekende of onbruikbare waarde.
11. D is een verzameling eisen die aangeven hoe een echte relationele database-querytaal er uit zou moeten zien. D is geen taal op zichzelf.
12. Tutorial D is een verschijningsvorm van D, die een echte relationele database-querytaal is. Tutorial D werd ontwikkeld om didactische redenen.

Er kan nog veel meer gezegd worden over het relationele model, zoals bijvoorbeeld de definities van predikaten en beweringen, zoals ik die hiervoor gebruikt hebt, waarbij ik een beroep doe op de intuïtieve perceptie van de lezer.

SQL ≠ Het Relationele Model!

De zorgvuldige lezer zal het zijn opgevallen dat ik in mijn overzicht van de geschiedenis van het Relationele Model nergens de datums noem met betrekking tot de SQL-taal en de release van zijn standaards in 1992, 1999 en 2003. Dat moet niet als geringschattend worden opgevat. De SQL-taal heeft haar eigen merites, niet in de laatste plaats dat in de standaards een compromis kon worden bereikt tussen grote invloedrijke partners en andere gezichtspunten en belangen. Ik zou hier Chris Date willen parafraseren: "SQL ≠ The Relational Model!" [2].

> De meeste dataprofessionals zijn gewend geraakt aan logische inconsistenties in SQL

Er bestaat een aantal verschillen tussen het Relationele Model en SQL. Bovendien zijn de SQL-implementaties van de verschillende software-leveranciers niet consistent ten opzichte van elkaar, waardoor een verschillend aantal afwijkingen ten opzichte van het Relationele Model ontstaat. De meest voorkomende verschillen zijn:

- Duplicatie van rijen. Een SQL-tabel kan meer dan één keer dezelfde rij bevatten. Het is voor dezelfde tuple niet mogelijk om meer dan eens voor te komen in een relatie;
- Kolomvolgorde. Een SQL-tabel heeft een vastgestelde kolomvolgorde. Het ordenen van de attributen van een relatie is onbelangrijk;
- Kolomloze tabellen. SQL-tabellen moeten minstens één kolom bevatten. Relaties kunnen geen attributen hebben – de alomgevreesde TABLE_DEE en TABLE_DUM;
- NULL-waarde. SQL-tabellen kunnen NULL-waarden bevatten. Het Relationele Model is gebaseerd op binaire logica en kent dus alleen WAAR en NIET WAAR. Daarnaast is het interessant om op te merken dat SQL NULL niet alleen gebruikt om een onbekende of onbruikbare waarde aan te geven, maar ook in een overvloed aan andere gevallen. Een voorbeeld: de som van een lege verzameling is NULL (in de betekenis van

'niets'), het gemiddelde van de lege verzameling is NULL (in de betekenis van 'onbepaald') en een left join kan NULL als resultaat geven (wat hier betekent dat er geen matching rij is in de rechter operand).

Tutorial D versus SQL

In deze paragraaf wil ik een paar voorbeelden van Tutorial D laten zien en de mogelijkheid introduceren dat een echte relationele querytaal in feite gemakkelijker te schrijven is dan SQL – om over elegantie nog maar te zwijgen.
Voor de voorbeelden gebruiken we de volgende relaties en tabellen, die leveranciers (suppliers) en producten voorstellen. Voor deze voorbeelden heb ik materiaal uit Hugh Darwen's cursussen [3] wat aangepast. U herkent vast de wereldbekende relaties en tabellen met leveranciers en producten:

```
VAR S REAL RELATION
{S# CHAR, SNAME CHAR, STATUS INTEGER, CITY CHAR}
                                          KEY{S#};
VAR P REAL RELATION
{P# CHAR, PNAME CHAR, COLOR CHAR, CITY CHAR}
                                          KEY{P#};
CREATE TABLE S
(S# CHAR PRIMARY KEY, SNAME CHAR, STATUS INTEGER,
                                       CITY CHAR);
CREATE TABLE P
(P# CHAR PRIMARY KEY, PNAME CHAR, COLOR CHAR,
                                       CITY CHAR);
```

Voorbeeld 1.
SQL
```
SELECT * FROM S;
```
Tutorial D
```
S;
```

Voorbeeld 2.
SQL
```
SELECT * FROM S WHERE STATUS = 20;
```
Tutorial D
```
S WHERE STATUS = 20;
```

Voorbeeld 3.
SQL
```
SELECT S#, SNAME, STATUS FROM S WHERE STATUS = 20;
```
Tutorial D
```
(S {S#, SNAME, STATUS}) WHERE STATUS = 20;
```

Voorbeeld 4.
SQL
```
SELECT CITY, COUNT(*) AS c_count FROM S GROUP
                                          BY CITY;
```
Tutorial D
```
SUMMARIZE S PER S {CITY} ADD (COUNT AS c count);
```

Voorbeeld 5.
SQL
```
INSERT INTO S(S#, SNAME, STATUS, CITY)
VALUES ('S6', 'Dave', 10, 'Chicago');
INSERT INTO S(S#, SNAME, STATUS, CITY)
VALUES ('S7', 'Bob', 20, 'Toronto');
```
Tutorial D
```
INSERT S RELATION {
TUPLE {S# 'S6', SNAME 'Dave', STATUS 10,
                                  CITY 'Chicago' },
TUPLE {S# 'S7', SNAME 'Bob', STATUS 20,
                                CITY 'Toronto' }};
```

Merk op dat het in Tutorial D mogelijk is om meerdere tuples in een statement in te voegen.

Voorbeeld 6.
SQL
```
SELECT * FROM S, P WHERE S.CITY = P.CITY;
```
Tutorial D
```
S JOIN P;
```

Het maakt wel degelijk uit

Illustere academici als Chris Date, Hugh Darwen en Fabian Pascal – om er maar een paar te noemen – voeren al jaren aan dat de on-relationele kwaliteiten van SQL een dataprofessional belemmeren in de uitvoering van zijn dagelijkse werk.
Soms genereert SQL resultaten die relationeel incorrect zijn. Andere keren produceert SQL inconsistente resultaten afhankelijk van het gebruikte SQL-dialect. Ik heb een aantal curieuze verschillen tussen Nederlands en Vlaams leren kennen. Tot mijn verrassing leverde mijn aandrang tot 'poepen' compleet verschillende uitkomsten in Amsterdam en Gent. Ik vind het echter een stuk minder amusant als een SQL-statement met een resultaat op de proppen komt dat relationeel gezien nergens op slaat. Nog minder leuk vind ik het als hetzelfde statement tot het ene resultaat komt in het ene SQL-dialect en tot een ander resultaat in een ander dialect. Ik wil graag in een voorbeeld laten zien waar een gegeven probleem kan worden opgelost met twee logisch equivalente statements. Worden de twee statements geïmplementeerd in SQL dan levert dat verschillende resultaten op. In dit voorbeeld gebruiken we de volgende tabellen:

A	
PRODUCT	LOCATION
1	AMSTERDAM
2	NULL
NULL	ROTTERDAM
NULL	NULL
3	UTRECHT

B	
PRODUCT	LOCATION
3	UTRECHT
NULL	ROTTERDAM
2	NULL
NULL	NULL

Er wordt ons gevraagd om alle rijen te vinden die zowel in A als in B voorkomen. We kunnen dit probleem oplossen op twee logisch equivalente manieren. Het is mogelijk het probleem op te lossen door de intersectie van A en B te vinden. Het bijbehorende SQL statement ziet er zo uit:

```
SELECT PRODUCT, LOCATION FROM A
INTERSECT
SELECT PRODUCT, LOCATION FROM B;
```

Op de meeste SQL-gebaseerde databases zal dit statement de volgende rijen vinden:

PRODUCT	LOCATION
2	NULL
NULL	ROTTERDAM
NULL	NULL
3	UTRECHT

Als alternatief is het mogelijk het probleem op te lossen door alle rijen in A te vinden als een identieke rij in B voorkomt, door het volgende statement uit te voeren:

```
SELECT A.PRODUCT, A.LOCATION
FROM A, B
WHERE A.PRODUCT = B.PRODUCT
AND A.LOCATION = B.LOCATION;
```

Op dezelfde relationele database waarop het eerste statement werd uitgevoerd, geeft dit statement als resultaat:

PRODUCT	LOCATION
3	UTRECHT

Twee relationele, logisch equivalente statements produceren in SQL twee verschillende resultaten, wat betekent dat SQL niet relationeel is – quod erat demonstrandum.

De meeste dataprofessionals zijn gewend geraakt aan dit soort logische inconsistenties in de SQL-taal. Velen van ons hebben door de jaren heen een aantal 'work-arounds' verzameld, om er zeker van te zijn dat het met een SQL statement verkregen resultaat ook de beoogde uitkomst is.

Consistentie is een van de belangrijkste kwaliteiten die absoluut gewaarborgd moeten zijn in een DBMS. Echter, ik heb met een

algemeen voorbeeld aangetoond waar SQL de consistentie verliest. Als een datamodel ook nog inconsistentie vertoont, zoals SQL doet, dan is het mogelijk om in dat datamodel te laten zien dat 1 = 0 [1].

Echte Relationele Databases

Andersom garandeert het Relationele Model de creatie van een logische representatie van informatie die consistent is. Echte Relationele Databases – die bestaan bij de gratie van Onechte Relationele Databases – waren tot voor kort nog niet zo algemeen beschikbaar (inclusief support) zoals vandaag de dag het geval is. Op dit moment wordt een aanzienlijk aantal Echte Relationele Databases aangeboden. Ik noem er enkele die absoluut de moeite waard zijn:

- **Aldat**, een implementatie van Relix en JRelix;
- **CsiDB**, een set van proprietary C++ library's;
- **Duro**, een set van gratis C library's;
- **Opus**, een set van gratis C library's;
- **Rosetta**, een gratis Perl-implementatie;
- **Dataphor**. De allereerste implementatie van D is D4, de taal van Alphora's Dataphor. Dataphor is een commercieel product dat de taal bovenop bestaande SQL-databases implementeert. Alphora heeft aangekondigd in het tweede kwartaal van 2008 groot nieuws over de toekomst van Dataphor bekend te maken. Op het moment van schrijven is geen verdere infor-matie beschikbaar;
- **Rel**, een gratis open source echt relationeel DBMS, geschreven in Java. Rel werd ontwikkeld door Dave Voorhis van de University of Derby en implementeert een groot gedeelte van Tutorial D. Het is mogelijk de laatste versie 0.0.13 Alpha van dit product te downloaden [4]. De installatieprocedure is snel, waardoor u in staat bent om binnen enkele minuten een werkende Echte Relationele Database op uw computer te hebben;
- **TransRelational Model**, een gepatenteerd ontwerp van Stephen Tarin, CEO van Required Technologies Inc. Het implementeert een kolomgebaseerd DBMS, dat door Chris Date als "Echt Relationeel" is bestempeld. Bovendien stelt Date dat het TransRelational Model hoogstwaarschijnlijk de meest significante stap vooruit is sinds de introductie van het Relationele Model door Edgar Codd.

We kunnen er wel degelijk iets aan doen

"People nowadays are managed – not governed." Jarenlang werd dat herhaald door Tony Benn, een van de langst zittende parlementsleden in de UK. Professionals in het algemeen zijn hun politieke macht al lang geleden kwijtgeraakt en hebben zeer beperkte zeggenschap – als ze dat überhaupt al hebben – in het politiek draaiend houden van het land. De meeste gilden zijn lang geleden verdwenen en met hen de invloed die de beroepsmatige groepen ooit hadden. Geschiedenis is het verhaal van de overwinnaar en beroepsmatige gilden zijn door industriële en kapitalistische groeperingen afgeschilderd als de moeder van alle kwaad. Het is echter onweerlegbaar dat de gilden de laatste beroepsmatige organisaties vormden die in staat waren op te komen voor de belangen van hun beroepsgroep.
Terwijl Hollywood's scenarioschrijvers met hun spieren rollen via hun Writers Guilds, zijn dataprofessionals niet als groep georganiseerd. Ze missen zo, naast vele andere zaken, de kans om hun invloed en overredingskracht uit te oefenen op de software-leveranciers. Desondanks is een toenemend aantal studenten aan de universiteiten nu in de positie om het Relationele Model aan den lijve te ervaren. Niet als studie-opdracht maar door hun studie in praktijk te brengen, gebruik makend van een Echte Relationele Database zoals Rel. Deze nieuwe dataprofessionals zullen ervaren dat er betere opties zijn dan SQL en zullen in de bevoorrechte positie verkeren om Echte Relationele Producten te eisen.

Literatuur

1. *Chris Date, Hugh Darwen: Databases, Types and the Relational Model: The Third Manifesto.*
2. *Chris Date: Database In Depth.*
3. *Hugh Darwen: http://www.dcs.warwick.ac.uk/~hugh/#CS252.*
4. *Dave Voorhis: http://dbappbuilder.sourceforge.net/Rel.html.*

Ivan Pellegrin

Ivan Pellegrin (ivanpellegrin@kvl.nl) is Principal Consultant en Principal Enterprise Architect bij KVL Inspiratie Technologie.