Het gaat over niets

NULL in het logisch en fysiek gegevensmodel

Toon Loonen en Nicolette Verdugt

In relationele systemen komt men weinig verkeerd gebruik van NULL tegen. Alleen zijn sommige ontwerpers of DBA's slordig bij het definiëren van de tabellen en geven alle kolommen de NULL (default) optie mee, terwijl enkele van deze kolommen wel verplicht en noodzakelijk zijn. De aspecten van het gebruik van NULL worden daarom eens op een rijtje gezet. Daarbij komen veel opvallende zaken naar voren, zoals syntax-verschillen tussen producten of, nog opvallender, verschillende resultaten bij dezelfde syntax.

In dit artikel worden de functionele aspecten en de implementatiedetails van het gebruik van NULL uitgewerkt. Dit wordt gedaan voor Oracle en Microsoft SQL Server. Sybase (Application Server Enterprice, ASE) is vanwege de gelijke oorsprong meestal gelijk aan MS SQL Server (maar niet altijd). Andere producten zullen gelijke functionele mogelijkheden bieden maar waarschijnlijk weer afwijken van de genoemde producten.

Wat is NULL

De attributen in het logisch gegevensmodel of de kolommen in de fysieke tabellen (voortaan kortweg de kolommen genoemd) krijgen de waarde NULL bij een van de volgende mogelijkheden:

- De waarde van de kolom is niet van toepassing op het betreffende object (bijvoorbeeld de huwelijksdatum bij ongehuwden, meisjesnaam bij ongehuwde vrouwen en bij mannen);
- De waarde is onbekend (het e-mailadres van een lid van de vereniging).

Soms kan een waarde bekend zijn maar toch niet ingevuld worden, omdat de gebruiker het niet nodig vindt voor zijn verdere werk met het systeem, bijvoorbeeld het hiervoor genoemde e-mailadres. Als de waarde niet is ingevuld, kan niet direct geconcludeerd worden om welke van bovenstaande redenen de kolom niet is ingevuld. De ANSI-standaard en de diverse producten maken namelijk geen onderscheid tussen NULL voor "niet van toepassing" en NULL voor "onbekend".
Voor numerieke velden is er vaak, maar niet altijd, een duidelijk verschil tussen de waarde 0 (in spreektaal ook vaak "zero") en NULL (in de spreektaal ook vaak "NILL"). Bijvoorbeeld: de temperatuur is gemeten en de waarde is 0 graden Celsius of de waarde is onbekend (meetapparatuur heeft geweigerd). Bij berekeningen voor de gemiddelde temperatuur moet de waarde "zero" wel meegenomen worden, de waarde NULL niet.
Bij systemen die gebruik maken van relationele databases komt men op dit punt weinig fouten meer tegen. Bij systemen die geen gebruik maken van relationele databases, maar bijvoorbeeld van C-ISAM of andere (indexed of ASCII) files, wordt dit aspect vaak niet correct geïmplementeerd.

Bij bedragen is het verschil vaak minder relevant. Als bij een factuur 0 (zero) Euro boete betaald moet worden of de boete is NULL (niet van toepassing), dan zal dat voor de debiteur geen echt verschil maken. Toch blijft het in alle gevallen gewenst om ook hier goed met het concept NULL om te gaan, bijvoorbeeld:

- Bij een nieuwe factuur is de boete NULL;
- Bij de eerste aanmaning is de boete 0 (zero);
- Bij de tweede aanmaning is de boete 10 Euro, etcetera.

NULL in character-kolommen

Bij tekstvelden van een beperkte lengte (type char() of varchar() in de database) moet ook goed onderscheiden worden of een kolom leeg mag zijn (de waarde NULL toegestaan) of niet. De meisjesnaam of het e-mailadres (zoals bovengenoemd) blijven leeg als ze onbekend of niet van toepassing zijn, anders staat er in de kolom een reeks tekens.
Deze reeks tekens kan eventueel 0 (zero) tekens bevatten dus een lege string. Als een string wordt omgeven door enkele quotes ('abc') dan wordt de *lege string* (string met 0 tekens) gerepresenteerd door twee enkele quotes na elkaar: ''.

> NULL geeft aan dat de kolom geen waarde bevat

Oracle heeft hier een afwijkende implementatie en verwerkt (althans tot versie 9i) een lege string als NULL. In de documentatie (Oracle9i SQL Reference) komt men echter de volgende tekst tegen: *"Oracle currently treats a character value with a length of zero as NULL. However, this may not continue to be true in future releases, and Oracle recommends that you do not treat empty*

strings the same as NULLs". Het is dus ook in Oracle belangrijk om geen lege string (dus '') te gebruiken in plaats van NULL met het oog op mogelijke toekomstige wijzigingen.
Het verschil tussen een lege string en NULL mag soms zuiver academisch zijn, maar er kunnen situaties zijn waarbij het onderscheid wel noodzakelijk is. Bijvoorbeeld als voor elk teken in de string een manipulatie nodig is (berekenen checkdigit in een bankrekeningnummer, character-conversie, etcetera).
In pseudocode:

```
Voor i = 1 t/m lengte van de string: Doe iets.
```

Een lege string heeft de lengte 0; Voor NULL is de lengte onbekend. Een FOR of IF conditie kan hier dus anders op reageren.

NULL in datum- en boolean-kolommen

Voor datums is het voorgaande ook van toepassing: de huwelijksdatum van een persoon is niet van toepassing of onbekend.
De datum dat een factuur betaald is, is niet van toepassing als de factuur nog niet betaald is. Het verschil in dagen tussen de factuurdatum en de datum van de betaling is in dit geval dus ook niet van toepassing, dus NULL.
Een boolean kan alleen de waarde Ja (= True of 1) of Nee (False of 0) hebben of onbekend (NULL) zijn. Het wordt aangegeven met het type bit in Sybase en Microsoft SQL Server; Oracle kent dit type niet. Alternatief is dan een char(1) kolom met de waarde 'J', 'N' of NULL. Voor een persoon kan de kolom Gehuwd Ja/Nee de waarde Ja of Nee hebben of onbekend zijn. Voor niet natuurlijke personen (BV of stichting) is de waarde zelfs niet van toepassing (dus altijd NULL).

> Soms zijn er regels te definiëren wanneer een attribuut wel of niet gevuld moet zijn

Let op: We praten hier steeds over "de waarde NULL", maar eigenlijk is dit niet correct. NULL geeft namelijk aan dat de kolom geen waarde bevat. We zullen in dit artikel het gewone spraakgebruik volgen en op dit punt niet te principieel zijn.

Condities in de programmatuur (of in het ontwerp) hebben altijd een waarde WAAR of ONWAAR (True of False). In de bekende RDBMS'en wordt bij condities geen onderscheid gemaakt tussen Onwaar en Onbekend: Onbekend wordt hier altijd op False gesteld. In feite is dit niet geheel correct en zou hier, net als hiervoor bij het kolomtype boolean, onderscheid gemaakt moeten worden tussen True, False en onbekend. Indien dit nodig is voor een toepassing zal het zelf geprogrammeerd moeten worden, bijvoorbeeld met:

Fuzzy logic

De logica met drie waarden (Waar, Onwaar of Onbekend) heet "Three-valued logic" (of "Three value logic"). Nog een stap verder komen we in de Fuzzy logic die veel bij Artificial Intelligence-systemen wordt gebruikt. Hierbij kan een bewering ook "een beetje waar zijn". Bijvoorbeeld: Onder de 10 graden Celsius is het koud. Boven de 20 graden is het warm. Bij 15 graden is de bewering: "Het is koud" half (50 procent of 0.5) waar.
Hiermee kan weer verder geredeneerd worden. Als twee beweringen A 25 procent waar en B 75 procent waar zijn, dan is:
- de bewering "A AND B" 25 procent waar (minimum A en B)
- de bewering "A OR B" 75 procent waar (maximum A en B).

Als we voor A en B alleen nog 0 of 1 mogen invullen moeten we onze bekende RDBMS-logica weer terug vinden.
Op internet is veel informatie over Fuzzy logic te vinden.
Zie bijvoorbeeld:
http://directory.google.com/Top/Computers/Artificial_Intelligence/Fuzzy/
http://www-2.cs.cmu.edu/Groups/AI/html/faqs/ai/fuzzy/part1/faq-doc-2.html
http://www.webopedia.com/TERM/f/fuzzy_logic.html

```
if status = 'J' then …
else if status = 'N' then …
else if status is NULL then …
```

NULL in het logisch model

Bij het ontwerp van het logisch gegevensmodel moet voor elk attribuut nagegaan worden of NULL wel of niet is toegestaan, met andere woorden, of het veld verplicht ingevuld moet worden of niet. Voor attributen die onbekend of niet van toepassing kunnen zijn, wordt NULL toegestaan. Ook voor attributen die wel bekend zijn maar waarvoor de waarde niet essentieel is voor verdere verwerking in het systeem, kan NULL (zeg maar "optioneel") worden toegestaan. De gebruiker kan dan zelf beslissen of hij de waarde wel of niet invult.
Soms zijn er regels (business rules of validaties) te definiëren wanneer een attribuut wel of niet gevuld moet zijn. Bijvoorbeeld:
- Als de status is "gehuwd", dan moet ook de huwelijksdatum ingevuld zijn;
- Als de status is "gehuwd" en geslacht is "vrouw", dan moet ook de meisjesnaam ingevuld zijn.

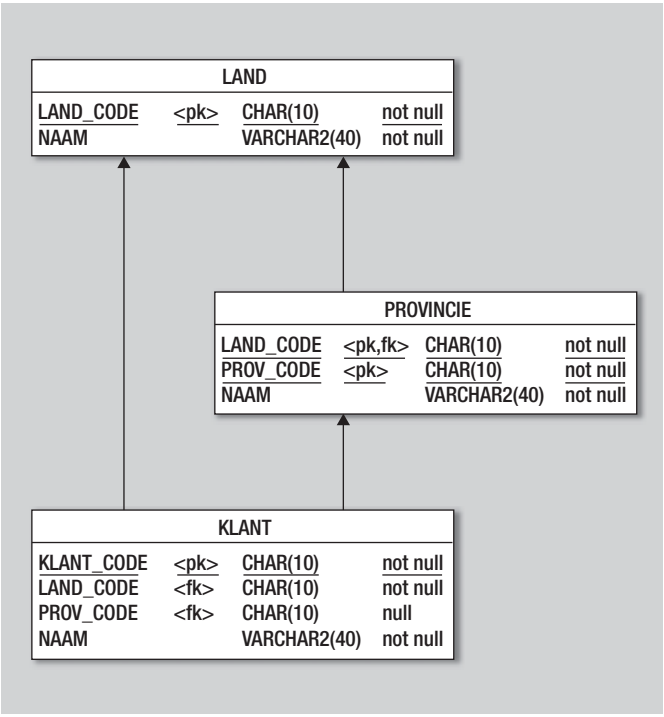
Deze regels kunnen niet met de simpele NULL toegestaan, worden gevalideerd. Wel kan men extra validatieregels in het gegevensmodel en in de database opnemen die hierop controleren. Een alternatief is het creëren van subtypes voor bijvoorbeeld gehuwde personen en vrouwen. Voor deze subtypes zijn deze attributen dan verplicht. In the supertype (persoon) komen deze attributen niet voor. Dit zal echter vaak een te grote stap zijn en leiden tot een onoverzichtelijk en versnipperd model[1]. Als er

echter veel bij elkaar horende attributen zijn, die samen vaak niet van toepassing zijn, dan kan men overwegen om deze groep attributen naar een eigen tabel af te splitsen met een 1 op 0,1 relatie met de oorspronkelijke tabel.

Null in primary en foreign keys

De (1 of meer) kolommen van de primary key zijn altijd verplicht, dus NOT NULL. Kandidaat sleutels (alternatieve sleutel, bijvoorbeeld postcode en huisnummer van een klant) en andere unieke attributen kunnen wel optioneel (leeg, NULL) zijn.
Een foreign key verwijst naar de primary key van een andere tabel. De waarde van de foreign key moet dus ook voorkomen in de gerelateerde tabel. Als de waarde van de foreign key kolom niet van toepassing is blijft deze leeg (NULL). Er is dan dus geen verwijzing naar de andere tabel; het is een optionele relatie.
Voorbeeld: de postcode in de klantentabel kan verwijzen naar een postcode in de postcodetabel (met alleen Nederlandse postcodes). Voor buitenlandse klanten/adressen zal deze postcode dan leeg moeten blijven en een alternatieve adresregel gevuld worden. Logisch gezien zijn dit weer twee subtypes met verschillende relaties[1].

De foreign key landcode in de klantentabel zal verwijzen naar de primary key landcode in de landentabel. Als echter een groot deel van de klanten in Nederland woont zou men kunnen afspreken om hiervoor de landcode leeg te laten. In dit geval heeft NULL een speciale betekenis, namelijk Land = Nederland. Alhoewel conceptueel onjuist kan het wel praktisch zijn. Een beter alternatief is om hier gebruik te maken van de default clause in het create of alter table statement:



Afbeelding 1: Relatie tussen de tabellen Land, Provincie en Klant.

```
create table klant (..,
land_code    char(6) default 'nl',
status_code  not NULL char(2) default 'xx');
```

Indien de kolom LAND_CODE in het insert statement wordt weggelaten krijgt deze de waarde 'NL' maar de kolom kan nog wel expliciet op NULL gezet worden met:

```
insert into klant (.., land_code, ..) values (..,
NULL, ..);
```

Voor de STATUS_CODE is dit niet toegestaan in verband met de NOT NULL clause.

Het schema in afbeelding 1 toont een wat complexere relatie tussen de tabellen Land, Provincie en Klant. Een Land kan optioneel opgedeeld zijn in Provincies (of districten enzovoort). Bij een Klant wordt aangegeven in welk Land hij woont en, indien van toepassing, in welke Provincie. De landcode is hier altijd verplicht, de provinciecode alleen als het betreffende Land is opgedeeld, ofwel als bij de Klant ook de Provincie bekend is.
De foreign key van Klant naar Provincie bestaat hier uit twee attributen, land_code en prov_code. Deze key is niet verplicht. Het eerste deel van deze foreign key (de land_code) is tevens de foreign key naar de tabel Land en deze is wel verplicht.
Deze situatie is wel een enkele keer voorgekomen en niet elk systeem (4GL of RDBMS) gaat hier mee om zoals het hier bedoeld is (Oracle 9i en SQL Server hanteren deze situatie goed).
In sommige gevallen zal het nodig zijn om twee gescheiden foreign keys te definiëren en zal helaas de landcode dubbel, dus redundant, in de klanttabel moeten worden opgenomen.
Het alternatief is om de relaties niet met een foreign key constraint te controleren maar bijvoorbeeld met een trigger of in de applicatie.
Als in bovenstaand voorbeeld met de relatie Land-Klant een land wordt verwijderd zijn er drie mogelijkheden:

- Een restricted delete: Dit mag alleen als er geen klanten meer zijn;
- Een cascaded delete: De klanten in dit land worden mee verwijderd;
- Een Set NULL delete: De foreign key naar het land wordt op NULL gezet.

In het volgende deel wordt een en ander uitgewerkt naar SQL query's.

Literatuur

1. Loonen, Modelleren van subtypes. Database Magazine 1999/1.

Toon Loonen en Nicolette Verdugt

Toon Loonen (toon.loonen@cgey.nl) en Nicolette Verdugt (nicolette.verdugt@cgey.nl) zijn werkzaam bij Cap Gemini Ernst & Young.