Niet elk gegeven is geschikt voor unieke identificatie

Logische en technische sleutels in gegevensmodel

Toon Loonen

Sommige gegevensmodelleurs of DBA's zijn de mening toegedaan dat aan elke tabel een technische sleutel moet worden toegekend als primaire sleutel. Dit ondanks een reeds aanwezige unieke functionele sleutel. Is dat inderdaad verstandig? Soms of altijd?

Deze vragen kunnen tot veel discussie leiden. Zie bijvoorbeeld de referenties 4 en 5. In dit artikel wordt antwoord gegeven op deze vragen. In het dagelijkse leven worden objecten (medewerkers, klanten, artikelen, orders) op een praktische, voor menselijke communicatie geschikte, wijze geïdentificeerd:
- medewerkers en klanten met hun naam;
- artikelen met hun naam en eventueel verdere detaillering zoals kleur, maat en uitvoering;
- orders met een aanduiding als "de order van klant X van datum D".

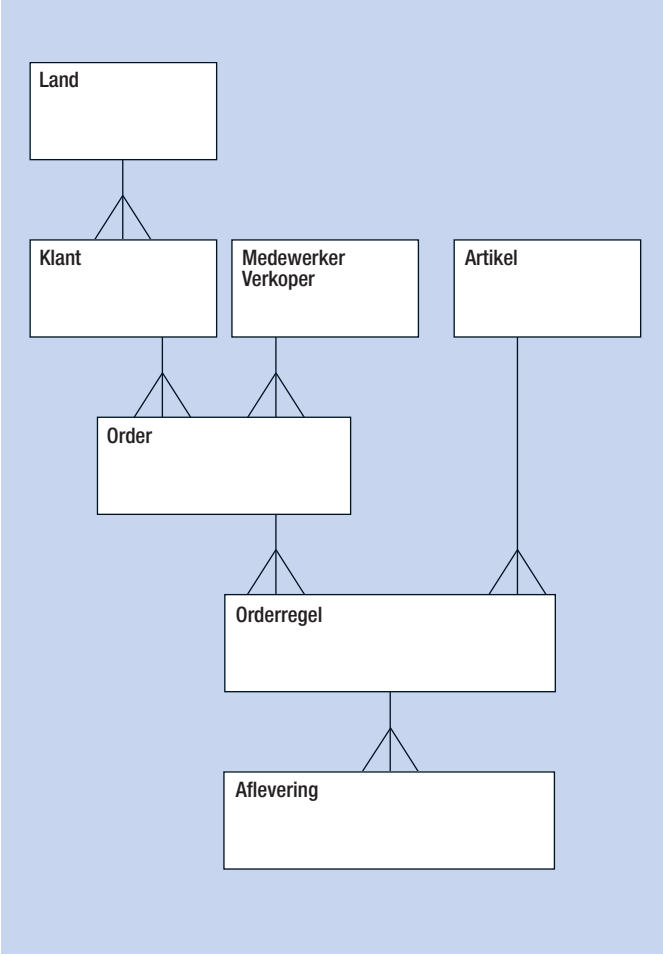
In het spraakgebruik gaat dit meestal (maar niet altijd) goed. Als een identificatie niet uniek is (er zijn twee medewerkers met dezelfde naam, een klant heeft op een dag twee orders geplaatst) dan kan dit snel gezien worden en in overleg wordt het goede object geselecteerd.

In databases moeten we exacter werken. Daarom zal een niet gegarandeerd uniek gegeven zoals een naam niet geschikt zijn om te identificeren, dus als (functionele) primaire sleutel. Andere redenen waarom een attribuut soms niet geschikt is om een object te identificeren, dus te fungeren als primaire sleutel, zijn:
- Een gegeven is niet altijd aanwezig/van toepassing, bijvoorbeeld een sofinummer bij een buitenlandse persoon (Zie [2]);
- Een gegeven kan gemakkelijk van waarde veranderen, bijvoorbeeld de naam van een persoon (bij huwelijk) of de naam van een bedrijf/klant;
- Een gegeven is erg lang en onhandig om in te tikken bij referenties en opvragingen, bijvoorbeeld de naam van een artikel. Dit komt vaak voor in combinatie met het voorgaande: een lange naam van een artikel met kleur, maat en uitvoering kan gemakkelijk een tikfout bevatten en dan is er een wijziging nodig om de tikfout te corrigeren.

Een primaire sleutel gebruiken we dus om een gegeven uniek te identificeren, net zoals de naam van een persoon als we zeggen; "Gaat Irma weer scheiden?" In sommige gevallen hebben we (functioneel) geen primaire sleutel nodig, namelijk als er niet geïdentificeerd hoeft te worden. Stel bijvoorbeeld, we hebben een batch (sequentieel bestand, al of niet in de database) met betalingsgegevens. Als ik een factuur van tien euro betaal door meteen twee betaalopdrachten te maken van vijf euro, dan is het, bij correspondentie met de crediteur later, lastig om precies aan te geven welke betaling van vijf euro ik bedoel. Maar als deze batch sequentieel (en in een transactie) wordt verwerkt is het niet altijd nodig dat er nog naar de individuele regels verwezen kan worden. In dat geval is geen unieke identificatie van de records in de batch nodig. Natuurlijk kan het technisch toch handig zijn om een volgnummer aan elke regel toe te kennen, maar noodzakelijk is dat alleen als er wordt verwezen.
Een ander voorbeeld is een tabel met bijvoorbeeld systeemparameters. Als in deze tabel altijd maar een record mag staan, dan is het identificeren van dat ene record gemakkelijk door gewoon naar de tabel te verwijzen. Toch geef ik (fysiek) zo'n tabel wel een primaire sleutel, een uniek numeriek volgnummer of ID, en geef daarbij (met een constraint) aan dat de waarde van deze ID altijd '1' moet zijn. Ik heb dan impliciet voorkomen dat er ooit een tweede record in de tabel kan worden geplaatst.
De objecten, die naar een ander object verwijzen, bevatten nu een 'buitensleutel'. Bijvoorbeeld een order bevat de identificatie van de klant als buitensleutel om aan te geven bij welke klant deze order hoort. Een orderregel bevat een ordernummer

De sleutel van een adres binnen Nederland

Een adres in Nederland kan uniek worden geïdentificeerd met de combinatie Postcode, Huisnummer en Huisnummer Toevoeging, Bijvoorbeeld 1234AB, 123, A.
Bij veel adressen is de Huisnummer Toevoeging echter niet van toepassing, in de database dus NULL. Een RDBMS (Oracle, MS SQL Server) zal echter eisen dat alle attributen van een samengestelde primaire sleutel verplicht (NOT NULL) zijn. Hier is deze combinatie dus niet geschikt als primaire sleutel en zal een technische of surrogaat sleutel aan de entiteit/tabel moeten worden toegevoegd.



Afbeelding 1: Het model van de gebruikte entiteiten.

en een artikelnummer om naar de betreffende order respectievelijk het artikel te verwijzen.

Wat is bruikbaar als identificerend gegeven

Als de naam van een medewerker of klant (persoon of bedrijf) niet geschikt is om als primaire sleutel te gebruiken, wat hebben we dan als alternatief? Voor een medewerker zou dit het personeelsnummer of het sofinummer kunnen zijn. Deze zijn altijd aanwezig, kort, en wijzigen in principe niet. Het personeelsnummer is een gegeven waarover we zelf meer controle hebben, omdat het door onszelf wordt uitgegeven. Dit geldt niet voor het sofinummer[1]. Ook kan het zijn dat een buitenlandse medewerker (Poolse aspergesteker) tijdelijk nog geen sofinummer heeft. Daarom is het personeelsnummer een beter gegeven voor de primaire sleutel. Dit is een volgnummer dat door de afdeling personeelszaken of door het systeem zal worden uitgedeeld. Juist omdat het een verder betekenisloos volgnummer is, is het geschikt als primaire sleutel. Immers, als een nummer geen verdere betekenis heeft (dan alleen om uniek te identificeren) zal er ook nooit een noodzaak zijn om het nummer te wijzigen. Voor een klant zal door het systeem of door de betreffende afdeling bij inschrijving een volgnummer worden uitgedeeld, het klantnummer. Dit is zeker geschikt als primaire sleutel. Een alternatief voor het klantnummer is een klantcode, bijvoorbeeld de eerste zes posities van de naam plus een volgnummer (jansse123) zoals we dat vaak bij de userid's bij een internetprovider zien. We gaan er dan vanuit dat deze code niet wijzigt als de naam van de klant wijzigt (hetgeen bij een scheiding een vervelende situatie kan opleveren).

Voor een nieuw artikel zal door de betreffende afdeling en meestal volgens een bepaalde systematiek een artikelnummer worden samengesteld. Ook dit artikelnummer voldoet aan bovenstaande voorwaarden om het geschikt te maken voor unieke identificatie. Sommigen beweren echter dat als er betekenis in een gegeven zit, dit gegeven niet meer geschikt is om te identificeren, omdat het dan gemakkelijk kan wijzigen. Voor orders kan een volgnummer over alle orders (het ordernummer, uitgedeeld door het systeem) worden gebruikt als identificerend gegeven. Voor de orderregel (repeterende groep artikelen die op een order besteld worden) wordt meestal een samengestelde sleutel gebruikt. Deze bevat het ordernummer en een volgnummer binnen de order. Een alternatief is de samengestelde sleutel van ordernummer en artikelnummer, ervan uitgaande dat een artikel maar één keer op een order mag voorkomen.

De orderregel is in feite een koppelentiteit, in het leven geroepen om de meer-op-meer relatie tussen order en artikel te 'normaliseren'. Bij deze koppelentiteiten kan altijd de combinatie van de twee sleutels van de twee bovenliggende entiteiten als sleutel worden gebruikt. Deze moet worden aangevuld met een volgnummer als deze combinatie niet uniek is, dus in ons voorbeeld: als een artikel twee keer op een order mag voorkomen.

De hierboven genoemde sleutels zijn functionele sleutels, de gebruiker zal ze gebruiken in zijn communicatie met andere personen of met het systeem:

- Het salaris van Jan, personeelsnummer 123, moet aangepast;
- Order 987654 (van klant XYZ, klantnummer 1234) is nog niet geleverd;
- Artikel 12AB3456CD78 op order 987654 (van klant XYZ, klantnummer 1234) is op en kan niet geleverd worden;
- Artikel 12AB3456CD78 moet bijbesteld worden.

Naast de primaire sleutel zijn er zoekargumenten: attributen waarop gemakkelijk naar een object kan worden gezocht maar die niet aan de strikte eisen van de primaire sleutel voldoen. Dit kunnen zijn:

- Bij een klant: naam, postcode of woonplaats;
- Bij een order: klantnummer (hiervoor opgezocht via postcode);
- Een artikel via een deel van de naam, artikelsoort, etcetera.

Soms zijn er twee attributen die elk geschikt zijn om als primaire sleutel gebruikt te worden, bijvoorbeeld bij medewerkers het personeelsnummer en het sofinummer. We kiezen dan één attribuut als de primaire sleutel en het andere attribuut wordt een 'kandidaat sleutel' en krijgt fysiek ook een unieke index. Volgnummers, zoals het genoemde klantnummer en ordernummer, zijn logische sleutels en geen fysieke sleutels omdat ze in

Genereren van een ID

Het genereren van een volgnummer voor een functionele of technische sleutel kan in de verschillende RDBMS-producten als volgt.

Oracle heeft een object type 'SEQUENCE'. Hiervan kan telkens de volgende waarde worden uitgevraagd en deze kan worden gebruikt om een volgnummer toe te kennen.

Informix heeft als kolomtype naast numeriek ook SERIAL.
Bij dit kolomtype zal Informix zelf hier een volgnummer in plaatsen.
Voorbeeld:

```
CREATE TABLE MYTABLE (ID SERIAL NOT NULL, ..)
```

SQL Server en Sybase kennen de property IDENTITY als toevoeging op bijvoorbeeld INT, waarbij ook automatisch een volgnummer wordt toegekend.
Voor SQL Server:

```
CREATE TABLE MYTABLE
                    (ID INT IDENTITY (10, 2), ..)
```

Waarbij 10 de startwaarde en 2 de increment is; default is (1,1).

DB2:

```
CREATE TABLE MYTABLE (ID INT NOT NULL
GENERATED ALWAYS AS IDENTITY
           (START WITH 10, INCREMENT BY 2), ..);
```

MySQL:

```
CREATE TABLE MY_TABLE (
ID INT AUTO_INCREMENT, ..,
PRIMARY_KEY(ID));
```

Bij alle systemen is het natuurlijk mogelijk om zelf een tabel met volgnummers te introduceren en bij elke insert de laatst gebruikte waarde uit te vragen, dit nummer 1 op te hogen en samen met de insert de transactie te committen. Dit zal een slechtere performance geven dan de hiervoor genoemde oplossingen, omdat het DBMS hiervoor geoptimaliseerde oplossingen heeft, bijvoorbeeld door een aantal (100, 1000) nummers te cachen. Het nadeel van het cachen is dat bij een crash van het DBMS er nummers verloren kunnen gaan.

Als er geen gaten mogen vallen in de nummering kan derhalve een cache size van 1 worden gebruikt, hetgeen weer nadelig is voor de performance. Als alternatief kan in Oracle na het opstarten van de server (en eventueel alleen bij een server crash is) de sequence opnieuw worden aangemaakt met het laatst gebruikte nummer als startwaarde.

Als het nodig is om een unieke code te genereren voor een toepassing over meer fysieke systemen heen, dan kan dit als volgt:

1. Geef elk van de systemen een volgnummer (01 t/m 99): V1;
2. Genereer op elk systeem een volgnummer als hierboven met een sequence of serial: V2;
3. Het universele volgnummer (binnen de toepassing) is dan V2*100+V1.

Er zijn ook algoritmes, vooral in de JAVA-wereld, die een 'universeel' unieke code genereren. Dit is een hexadecimale code van 32 tekens. Voor een primaire sleutel is dit erg lang. Bedenk dat hiermee de index groot wordt en dus trager. De hiervoor genoemde numerieke sleutel zal meestal minder dan 10 bytes per record nodig hebben tegenover 32 bytes voor dit algoritme.

het ontwerp van functies, schermen en rapporten (factuur) genoemd worden en door gebruikers gebruikt worden. Zo zal de klant, bij het betalen, zijn klantnummer en ordernummer meegeven. Omdat een volgnummer gemakkelijk verkeerd ingetikt kan worden, is er een checkdigit aan toe te voegen, zoals bij banknummers en sofinummers (Zie [3]).

Technische sleutel naast functionele sleutel

Wanneer is in het fysieke ontwerp is nog een technische sleutel nodig naast de functionele sleutel? Als de functionele sleutel aan alle voorwaarden voldoet, (dus: gegarandeerd uniek is; altijd ingevuld is (niet NULL kan zijn, zie [2]); niet zal wijzigen; niet onhandig lang is) dan is er geen reden om nog een technische sleutel in het fysieke gegevensmodel op te nemen. Redenen om bij enkele tabellen toch een technische sleutel te introduceren zijn:

- Een code wordt functioneel als niet wijzigbaar beschouwd, maar bij het technisch ontwerp wordt er toch rekening mee gehouden dat deze kan wijzigen. Bijvoorbeeld de landcode in de tabel Land: Landcode en naam ZR Zaïre wordt CD Congo (zie http://publications.europa.eu/code/nl/nl-5000600.htm);
- De logische sleutel bevat erg veel attributen. De entiteit Aflevering bevat de samengestelde sleutel Ordernummer, Regelnummer, Afleveringnummer. Ik ben in zo'n geval al een keer op een samengestelde sleutel van zes attributen gestuit. In zo'n geval kan het nuttig zijn om ergens in deze hiërarchie een fysiek volgnummer te introduceren als primaire sleutel;
- De logische sleutel is om andere redenen niet handig als fysieke primaire sleutel.

Deze technische sleutels worden niet genoemd in het functioneel ontwerp en komen (in principe) ook niet op schermen of rapporten. Alleen een beheerder of DBA zal deze sleutel gebruiken bij het uitzoeken van een probleem. Als er voor een technische sleutel wordt gekozen als primaire sleutel in plaats van de functionele sleutel, dan zal in de verwijzende entiteiten ook de waarde van de technische sleutel worden opgenomen in de buitensleutel. Telkens wanneer bij het toevoegen van een

technische ID op een tabel er twee unieke indexen op deze tabel gemaakt worden, moet je je afvragen: heb ik deze technische sleutel wel nodig?
Ook zijn er (DBA- of ontwikkel-) tools die vereisen dat elke tabel een numerieke primaire sleutel (of tenminste een unieke, niet samengestelde numerieke index) heeft, bijvoorbeeld Hibernate. Als zo'n tool gebruikt wordt en noodzakelijk is, dan is het toevoegen van een technische sleutel dus (helaas?) onvermijdelijk, dit ondanks de nadelen van een extra index.
Daarnaast zijn er collega gegevensmodelleurs die van mening zijn dat elke tabel in ieder geval een surrogaatsleutel nodig heeft als er niet al een numerieke (en niet-samengestelde) functionele sleutel is, zie [4] en [5].

Performance-aspecten

Als de gegevens in de tabel op volgorde van de sleutel worden opgeslagen (Oracle: index organized table; Sybase en SQL Server: Clustered index), dan zullen deze gegevens ook fysiek bij elkaar staan in de tabel. Dit betekent dat voor het ophalen van bijvoorbeeld alle orderregels van een order (met primaire sleutel ordernummer en orderregelnummer) vaak maar één of enkele fysieke blokken van de tabel gelezen worden. Dit betekent minder I/O-verkeer en is dus sneller.
Als er in het gegevensmodel een landentabel geïntroduceerd wordt, dan kunnen we een ISO-Code (twee tekens, bijvoorbeeld NL voor Nederland) voor het land als logische sleutel gebruiken en een volgnummer (ID) als fysieke sleutel. De tabel bevat nu de volgende kolommen:
ID, ISO_Code, Naam.

> De orderregel is in feite een koppelentiteit

Om de ISO_Code van het land bij de klantgegevens te tonen is nu extra I/O naar de landentabel nodig. Als de ISO_Code zelf was gebruikt als sleutel, dan was deze ook als buitensleutel in de klanttabel opgenomen en was er voor het tonen van deze code bij de klantgegevens geen extra I/O naar de landentabel meer nodig.

Het moge ook duidelijk zijn dat het toevoegen van een technische sleutel aan een tabel naast de unieke logische sleutel meer diskopslag vereist en vertragend werkt bij het toevoegen en verwijderen van de gegevens, omdat er twee indexen worden bijgewerkt in plaats van een.
Een lange primaire sleutel, bijvoorbeeld een uitgebreide samengestelde sleutel of een artikelnaam, is ook slecht voor de performance, omdat elk blok van de indextabel maar weinig index entry's bevat en er dus meer blokken doorlopen moeten

worden om de gewenste gegevens op te zoeken. Hier zal het toevoegen van een extra ID als technische sleutel dus wel beter zijn voor de performance.

Conversie-aspecten

Als een primaire sleutel toch wijzigt is er meestal een (beperkte) conversie nodig. Bijvoorbeeld:
zou de ISO_Code gekozen zijn als (functionele en technische) sleutel voor de landentabel en er wijzigt toch een land van naam en code (zoals ZR Zaïre wordt CD Congo), dan moet:
- in de landentabel één record worden aangepast;
- in alle tabellen waar deze ISO_Code als buitensleutel gebruikt wordt, de oude code gewijzigd worden in de nieuwe code.

Conclusie

Objecten in een database, zoals klanten, orders en artikelen, moeten uniek geïdentificeerd worden om ernaar te kunnen refereren, zowel in de conversatie tussen personen als in de communicatie tussen een persoon en het systeem en ook intern binnen het systeem. Hiervoor kan soms het begrip gebruikt worden waarmee de gebruiker gewend is om over dit object te praten, zoals een artikelnummer. In veel gevallen moet er logisch een code of nummer (volgnummer, uit te delen door een afdeling of door het systeem) aan een begrip worden toegekend om het uniek te identificeren en erover te kunnen praten, zoals een personeelsnummer, klantnummer of ordernummer.

In een enkel geval kan een DBA besluiten om naast de functionele sleutel ook een technische sleutel te introduceren, bijvoorbeeld omdat:
- hij vermoedt dat de waarde van de functionele sleutel toch vaker wijzigt dan officieel verwacht wordt;
- hij de bestaande logische sleutel te onhandig vindt;
- de gebruikte tools (bijvoorbeeld Hibernate) dit noodzakelijk maken.

Door het juist selecteren van de primaire sleutel kan ook performancewinst worden behaald.

Noot

1. *Dit wordt vervangen door het burgerservicenummer, maar dat heeft weinig invloed op de waarde ervan voor de individuele medewerker.*

Literatuur

1. *Loonen. Gegevensmodel van een distributed data dictionary. Database Magazine 1997/4.*
2. *Loonen. NULL in het logisch en fysiek gegevensmodel. Database Magazine 2004/1,4.*
3. *Loonen. Kwaliteit van gegevens begint bij het begin. Database Magazine 2005/4,5.*
4. *http://immike.net/blog/2007/08/14/database-design-choosing-a-primary-key/.*
5. *www.agiledata.org/essays/keys.html.*

Toon Loonen (toon.loonen@capgemini.com) is werkzaam bij Capgemini.