

Het is altijd fijn om je tijd vooruit te zijn

De Polis Papers (1): Let's Make History

René Veldwijk en Frido van Orden

De Polisadministratie (PA) is in de kern een zeer eenvoudige database en kan worden weergegeven in een even eenvoudig ER-diagram, zie afbeelding 1. Er lijkt sprake te zijn van veel data en weinig structuur. Toch bestaat het Oracle-schema uit meer dan 100 tabellen en dat heeft voor een groot deel te maken met het 'total recall' karakter van de PA.

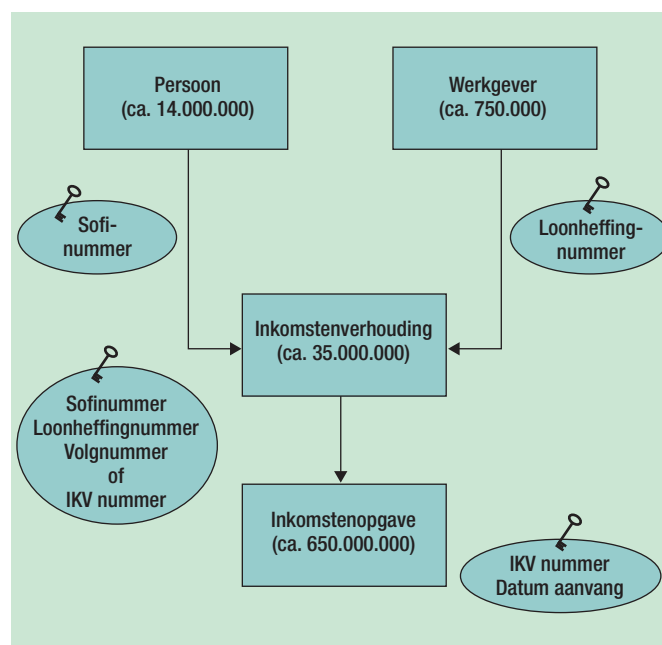
De 14 miljoen personen komen niet uit de Bevolkingsadministratie (GBA), zoals men zou kunnen verwachten, maar uit de maandelijkse of vierwekelijkse loonaangiften van 700.000 werkgevers. Op kinderen zonder vakantiebaantje na vinden we bijna heel Nederland terug, aangevuld met buitenlanders die hier economisch actief zijn. Nederlandse burgers hebben een burgerservicenummer (BSN) en de overigen hebben een sofi-nummer. Die nummers zien er hetzelfde uit en overlappen niet, dus iedereen in de PA is eenduidig te identificeren. Werkgevers – formeel: 'inhoudingsplichtigen' – identificeren zichzelf met een door de Belastingdienst toegekend Loonheffingnummer (LHNR). Dan komt het centrale entiteitstype van de PA: de relatie tussen een persoon en een werkgever ofwel de inkomstenverhouding (IKV). Een IKV wordt logisch geïdentificeerd door een BSN/sofinummer plus een LHNR plus een volgnummer. Dat laatste is nodig omdat iemand meer dan één verhouding met één werkgever kan hebben, ook op hetzelfde moment. Dat het aantal IKV's zo groot is komt doordat de arbeidsmarkt heel dynamisch is (denk aan vakantie- en uitzendwerk) en omdat ook alle uitkeringen (AOW, WW, WAO/WIA, pensioenen en lijfrentes) in de PA zitten. Er is helaas nog een derde reden: doordat het volgnummer aanvankelijk ongelukkig was gedefinieerd en doordat salarispakketten er onhandig mee omgaan, zijn er veel 'verknippte' inkomstenverhoudingen ontstaan. Echt schadelijk is het niet, maar het is natuurlijk heel onprettig dat het kernobject van een basisregistratie niet stabiel is.

Mede daarom hangen we de registratie van IKV's en aanverwante objecten niet op aan de gammele externe sleutel, maar kennen we zelf een unieke *id* toe. Hoe dan ook: de PA kent op dit moment ruim 18 miljoen actieve inkomstenverhoudingen, waarvoor maandelijks ruim 600.000 werkgevers aangifte doen. Na drie jaar aangiften verwerken kom je dan uit op een aantal inkomstenopgaven dat de half miljard riant overschrijdt. De tabel

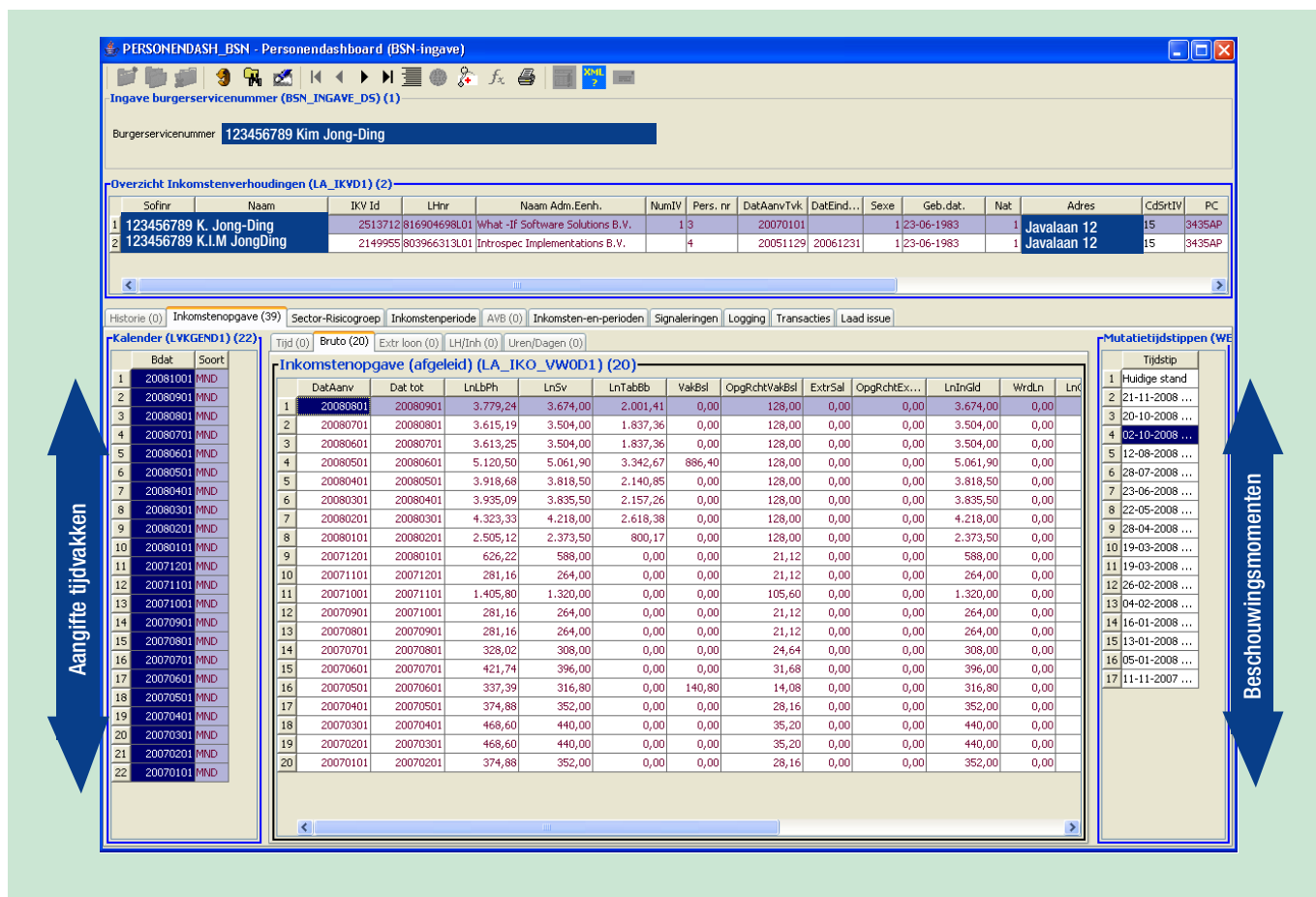
met inkomstenopgaven is inclusief indexen meer dan een halve Terabyte groot en dat is ongeveer de helft van de omvang van de hele PA database.

Groot of niet zo groot, dat is de vraag

De omvang van de PA database is als het schema niet goed is opgezet een *killer*. 'Omvang' is meer dan aantallen Gigabytes. Het is ook de verdeling van die Gigabytes over de tabellen, de volatiliteit van de geregistreerde gegevens en de aard en intensiteit van gegevensraadplegingen die er toe doen. Vanuit elk opzicht bekeken is de PA groot en als alles wordt gecombineerd reusachtig. Toch is de omvang van de PA uiteindelijk niet het probleem. De PA database blijkt vooralsnog zelfs zonder DBA hoogstandjes zoals partitionering en snapshot views te managen. Dat zegt iets over de stand van de techniek, maar zoals altijd staat of valt techniek met de manier waarop ermee wordt omgegaan door ontwerpers. Hoe belangrijk dat is blijkt als we de PA vergelijken met twee systemen met dezelfde inhoud. Zo bestaat de oorspronkelijke Polisadministratie nog steeds voor *fallback* doeleinden. Met precies dezelfde gegevensinhoud is dat ding tien keer groter. Dit enorme verschil is terug te voeren op



Afbeelding 1: ER diagram van de Polisadministratie.



Afbeelding 2: Twee tijdsdimensies – geschiedenis gezien vanuit het verleden.

enkele ongelukkige ontwerpkeuzes, niet op de database technologie, want die is dezelfde – Oracle. En dan is er ook nog een datawarehouse van meer dan 6 Terabyte. Ook stermodellieren blijkt als het om grote databases gaat een heel dure hobby. Omdat minder opslag leidt tot betere prestaties, kan de PA zijn eigen warehouse zijn en staat het datawarehouse nu op de nominatie om te worden geëlimineerd. Kortom, in echt massale omgevingen is de kwaliteit van gegevensmodellering van doorslaggevend belang voor performance, kosten en uiteindelijk haalbaarheid van een systeem.

Total Recall ...

Als we met dank aan de hardware en RDBMS-techniek en basale ontwerpvaardigheden de eerste hobbel eenmaal hebben genomen, dan komen we bij het echte probleem: de Polis-administratie is een *total recall* database. Dat wil zeggen dat mutaties op de PA nooit gegevens mogen overschrijven. Nog anders gezegd: elke toestand van de database is te reconstrueren voor elk moment in de tijd. De hoofdreden om dit te eisen is dat een basisadministratie in staat moet zijn om *mutatieverstrekingen* te doen. Dat kan alleen als elke historische toestand van gegevens bekend is. Iedere dag komen er bijvoorbeeld nog correcties binnen op loongegevens uit 2006. Die correcties zouden moeten worden doorgegeven aan alle afnemers die

eerder de verkeerde gegevens hadden ontvangen. Let wel: veel afnemers zijn daartoe nog niet in staat omdat ze zelf ook met tijdproblemen worstelen. Aan het einde van dit artikel komen we daarop nog terug.

Tijd-in-database is een specialisme van de auteurs. Tien jaar geleden hebben we daarom over dit onderwerp een serie artikelen geschreven. Het boekje dat DB/M daarover heeft uitgegeven is sinds kort digitaal beschikbaar op www.dbm.nl. In dat boekje kunt u lezen waarom tijd – en dan vooral tijd in twee dimensies – zo'n enorm probleem is. Laten we hier eens met de deur in huis vallen en een schermafbeelding bekijken, zie afbeelding 2.

Het scherm toont de geanonimiseerde gegevens van één van onze collega's. We zien zowel diens officiële naamgegevens (uit de GBA) als de gegevens die zijn aangeleverd door twee werkgevers. De PA kent voor deze persoon twee opeenvolgende inkomstenverhoudingen. De huidige werkgever, What-If Software, doet volgens de linkerkolom sinds begin 2007 maandelijks aangifte. De laatste aangifte was over de maand oktober 2008. Het centrale gegevensblok toont voor deze maanden de loongegevens. Tot zo ver niets ongewoons. Maar nu komt het: in de rechterkolom worden de tijdstippen getoond waarop er iets met de inkomensgegevens is gebeurd. Het scherm toont de situatie zoals die bekend was nadat er op 2 oktober een loonaangifte was verwerkt. Door te browsen in de lijst met mutatietijdstippen

reizen we als het ware door de tijd en zien we wat de PA op steeds andere momenten in de tijd had geregistreerd. In de linkerkolom kunnen we zien dat er tot en met de maand oktober is geleverd, maar dat is kennelijk gebeurd ná het beschouwingsmoment op 2 oktober, dus die gegevens worden niet getoond. Wellicht zijn de gegevens die wel worden getoond ook op een later moment gewijzigd.

We kunnen ook kijken naar één specifiek tijdvak vanuit diverse mutatiemomenten. Het scherm in afbeelding 3 laat zien dat het loon over januari 2008 op 26 februari werd aangeleverd en op 19 maart werd gecorrigeerd. De essentie is dat er steeds sprake is van twee onafhankelijke tijdsdimensies: een met de kalender-tijdvakken en een met de mutatiemomenten. De combinatie van deze twee dimensies resulteert in een database die nooit iets vergeet: *total recall*. Om afnemers op maat te kunnen bedienen is nog veel meer nodig, maar dit is de basis van alles. De manier waarop dit alles gemodelleerd is, is op enkele details na beschreven in het genoemde tijd-in-database boekje.

... in het kwadraat

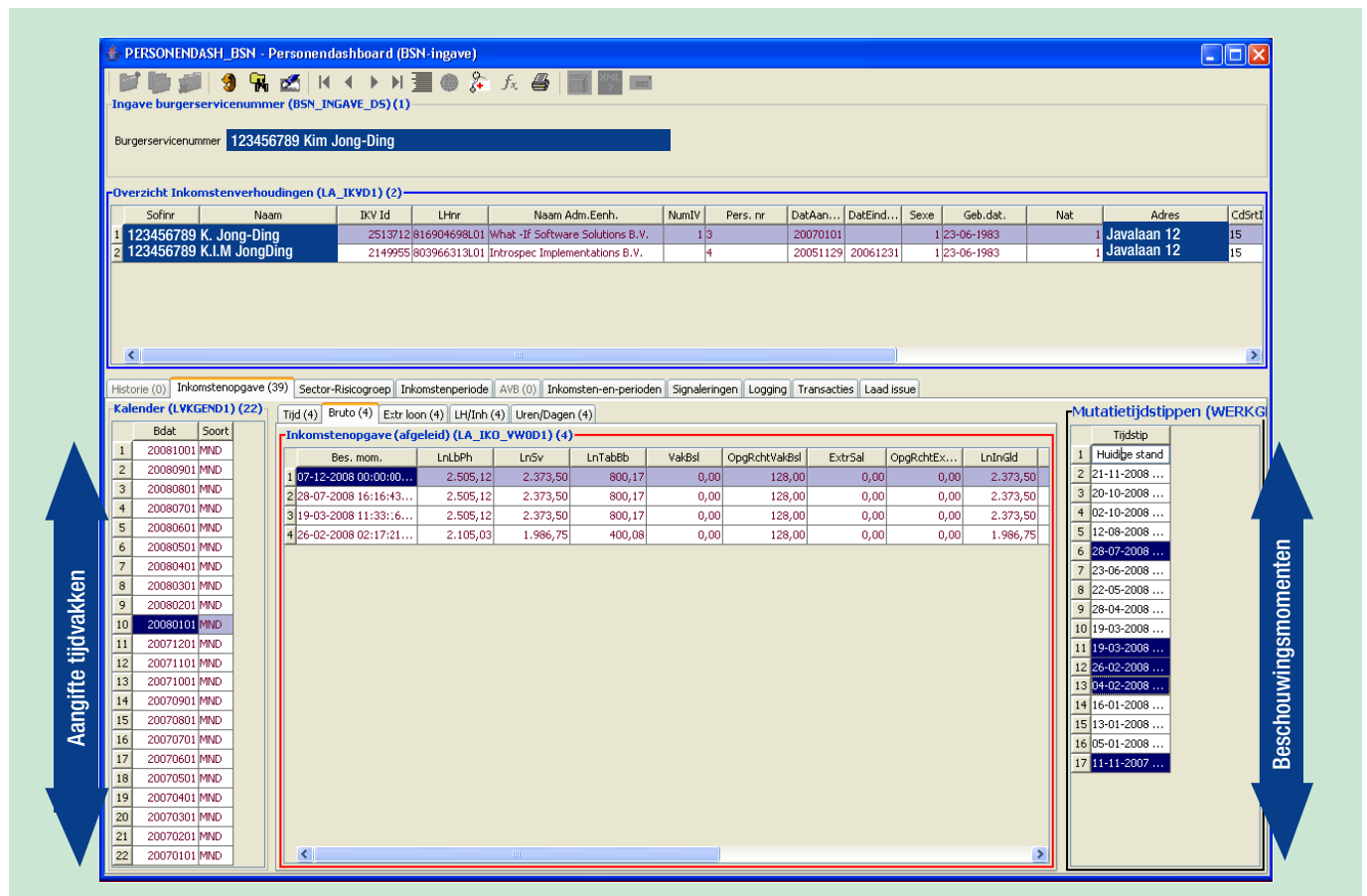
Een total recall database mag dan complex zijn, als die er eenmaal staat dan blijkt weer eens dat elk nadeel zijn voordeel heeft. Zo is het proces waarmee XML berichten worden geladen hypercomplex en dus foutgevoelig. Een *total recall* database

zoals de PA toont echter alle mutaties, dus ook de foute.

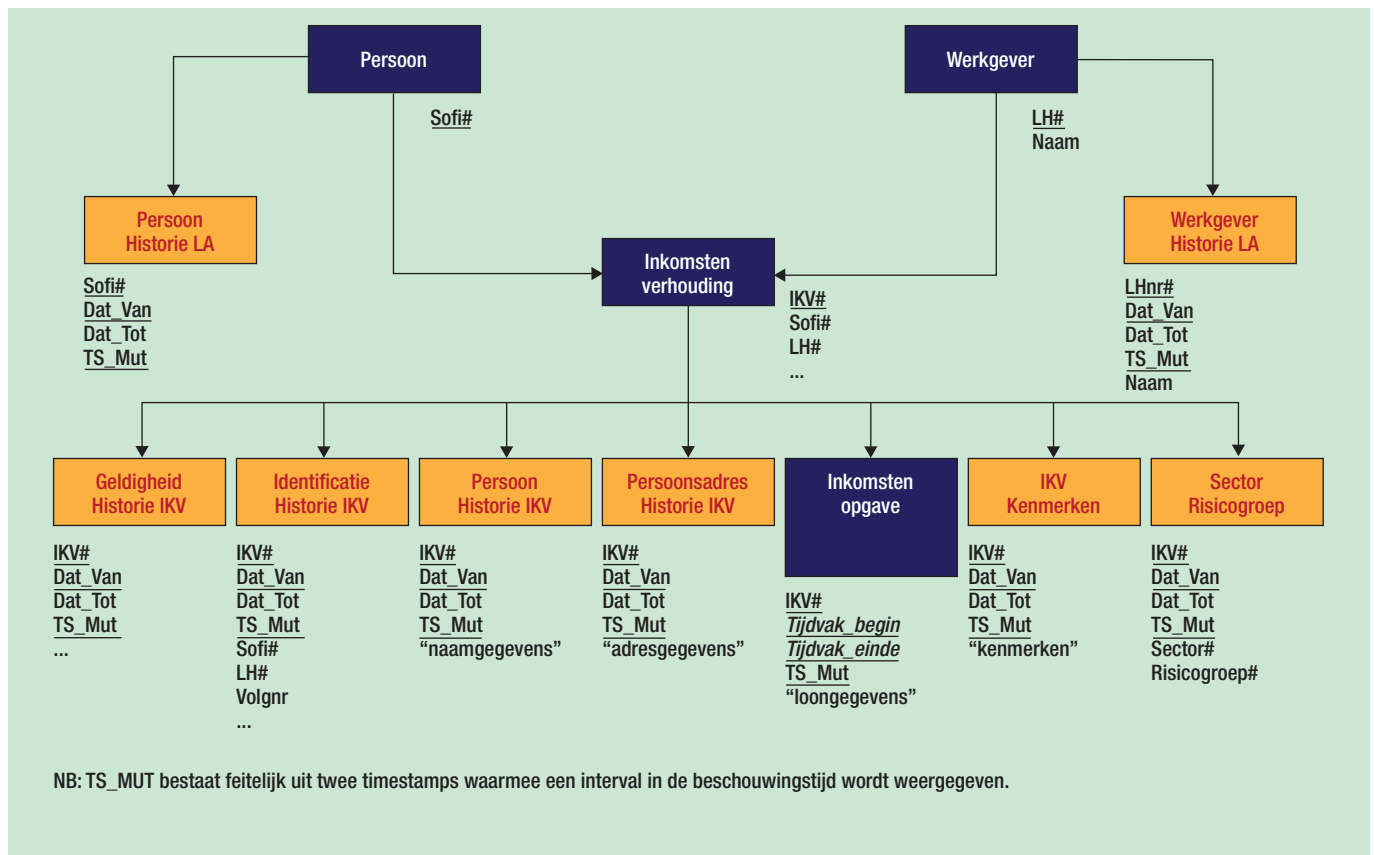
Correcties van fouten leiden net als alle andere mutaties niet tot overschrijvingen maar tot toevoegingen. Alles, ook programmeerfouten, is traceerbaar. Het bewaken van laadprocessen en verbeteren van laadfouten is dus relatief eenvoudig. In een extreem omvangrijke database gevuld met gevoelige gegevens die complexe transformaties hebben ondergaan, is zoiets van onschatbare waarde. Het is zelfs zo belangrijk dat recent de eis is gesteld dat ook de omgekeerde weg moet kunnen worden bewandeld: vanuit de standen in de PA moeten de XML berichten kunnen worden gereconstrueerd. Dat is een goed idee, want zo wordt het complexe laadproces controleerbaar en wordt het straks bijvoorbeeld mogelijk om de werkgevers terug te melden welke informatie de PA bevat in een hun bekend XML formaat. De combinatie van volledig historische vastlegging en volledige traceerbaarheid van verwerkte berichten biedt uitzicht op volledige greep op de complexe gegevenslogistiek en levert een overtreffende trap van traceerbaarheid op.

Het datamodel

Laten we eens kijken naar een iets verder uitgewerkte versie van het gegevensmodel waarmee we dit artikel begonnen, zie afbeelding 4. We zien de vier kern-objecttypen weer terug, maar nu in historische context. Allereerst valt op dat met uitzondering van



Afbeelding 3: Twee tijdsdimensies – kijken naar mutaties door te reizen in de tijd.



Afbeelding 4: Total recall kernmodel van de PA.

“Inkomstenopgave” de andere drie kern-objecttypen nog steeds zonder historie bestaan. “Persoon” bevat sofinummers die bestaan of eens hebben bestaan. “Werkgever” bevat het loonheffingnummer en de huidige werkgeversnaam. Wannéér personen en werkgevers in de loonaangiften precies bestonden, wordt geregistreerd in de onderliggende tabellen “Persoon-historie Loonaangifte” en “Werkgeverhistorie Loonaangifte”.

Ook stermodelleren blijkt als het om grote databases gaat een heel dure hobby

Bij werkgevers vinden we ook eventuele eerder gebruikte naamgegevens. Cruciaal is dat beide tijdlijnen terug komen in de sleutel van de historietabellen. Stel dat een werkgever een medewerker opgeeft en die opgave daarna weer intrekt, dan ziet de PA deze persoon niet voor het beschouwingmoment “nu” maar wel voor alle beschouwingmomenten tussen de opgave en de intrekking. Het entiteitstype “Inkomstenverhouding” is volledig opgedeeld in een aantal historietabellen die samenhangende groepen van gegevenselementen bevatten. We zien een historietabel die speciaal bedoeld is om de intervallen in de twee

tijdsdimensies te registreren gedurende welke een inkomstenverhouding volgens de werkgever bestaat. Daarna zien we een tabel waarin veranderingen van sleutelwaarden zijn opgenomen. Iemands sofinummer kan soms wijzigen en loonheffingnummers wijzigen regelmatig. De PA moet daar tegen kunnen. Vervolgens zien we twee historietabellen met persoonsgegevens die de werkgever meeleverd in de loonaangifte. En tenslotte zien we nog twee tabellen met allerlei codes en indicatoren waarmee een inkomstenverhouding wordt beschreven. Al deze samenhangende gegevensgroepen zijn in de beide tijdsdimensies gehistoriseerd. Het gevolg daarvan is dat het entiteitstype “Inkomstenverhouding” helemaal is uitgekleed en zelfs uit het schema kan worden verwijderd. Dat gaat echter te ver: we handhaven een fysieke tabel “Inkomstenverhouding” om nog een beetje greep te houden op de structurele integriteit van het gegevensmodel. Zo willen we bijvoorbeeld graag de verwijzende sleutels ook fysiek implementeren. Er is nog een tweede reden: de vele gegevensverstrekkingen die betrekking hebben op actuele gegevens willen we niet steeds reconstrueren vanuit een veelheid van historietabellen. Dat kost teveel performance en daarom kiezen we voor beheerste redundantie: alle actuele gegevens zijn te vinden in de tabel “Inkomstenverhouding”. Een logisch gevolg is wel dat het bij ingetrokken aangiften kan voorkomen dat we records vinden die op een primaire sleutel na alleen maar *null* waarden bevatten. Van groot belang is dat voor al deze tabellen zoveel mogelijk

wordt gewerkt met intervalverlenging. Uw werkgever levert bijvoorbeeld elke maand uw NAW-gegevens aan en doorgaans veranderen die niet. Is dat het geval dan doen we ook niets in de historietabellen. Behalve in de tabel "Identificatiehistorie IKV" staan voor niet afgesloten gevallen alle einddatums op 31/12/9999. Verandert er niets dan wordt er ook bijna niets gemuteerd en zo blijft de omvang van de database relatief klein. Dat doen we echter niet bij de inkomstenopgaven. Deze worden per tijdvak steeds opnieuw geregistreerd, ook wanneer er niets verandert. De reden daarvoor is dat loongegevens niet zoals de andere gegevens *voorraad*gegevens zijn maar *stroom*gegevens. We zien iemands loon van 2.000 euro over januari als iets anders dan dezelfde 2.000 euro over februari. Het alternatief is het registreren van een bedrag van 2.000 euro over de periode januari t/m februari. Afhankelijk van de vraagstelling moet de informatie dan worden geïnterpreteerd als 2.000 euro per maand of als 4.000 euro over twee maanden. Nee bedankt, dan maar liever een heel grote tabel.

Historie tot aan het gaatje?

Wie de voorschriften in het tijd-in-database boekje vergelijkt met de gemaakte keuzes bij het ontwerp van de PA ziet dat we ons recept voor historische databases niet volledig hebben overgenomen. De kern van onze theoretische benadering is dat historie voor de programmeur transparant wordt. Hier zou dat betekenen dat de programmeur alleen de vier tabellen in afbeelding 1 zou hoeven te kennen en de rest een kwestie is van parametriseren. Waarom hebben we ons eigen recept niet gebruikt? Gewoon omdat we het niet aandurfden op een database met dergelijke karakteristieken. En daarbij hebben we de ondersteuning van tijdlijnen nog onvoldoende geborgd in onze nieuwe tooling. In de beperking toont zich de meester, zullen we maar zeggen. Voor wie afbeelding 4 met afbeelding 1 vergelijkt en kennis neemt van de benadering in het tijd-in-database boekje zal het duidelijk zijn dat er nog een enorme reductie van complexiteit mogelijk is.

Ook in een ander opzicht zijn we nog niet klaar met historie. In komende artikelen zullen we de twee tijdlijnen steeds weer zien terugkomen. Zoals ook beschreven in het tijd-in-database boekje: als het verschijnsel tijd de grondslag vormt van een systeem dan wordt de hele database ermee besmet. Anders dan hier zal dan de nadruk liggen op de enorme voordelen die het werken met een *total recall* database oplevert.

De sterkste schakel in de keten?

Laten we aan het eind van dit eerste artikel eens als burgers kijken naar de PA database. Waarom wilde de overheid al die inkomstgegevens maandelijks op voorraad houden in één hollebolle database? De reden heet lastenverlichting: verzamel eenmalig de loongegevens en hergebruik deze maximaal. De PA is een basisregistratie zoals de GBA, het Kentekenregister en de Basisadministratie Gebouwen die nu wordt ontwikkeld. En inder-

daad neemt het gebruik van de gegevens van de PA sinds het *live* gaan begin 2008 razendsnel toe. Maar met al die afnemers is het cruciaal dat intelligent en eenduidig wordt vastgesteld wat de precieze betekenis is van de gegevens in de PA. En dan blijkt dat het probleem van tijd-in-database ook buiten het PA domein speelt. Neem bijvoorbeeld iemands bruto salaris van 2.000 euro over februari. Wat een bruto salaris is, is natuurlijk goed gedefinieerd, maar dan komt ook hier het tijdprobleem om de hoek kijken. Gaat het om 2.000 euro die in februari zijn *betaald* of om 2.000 euro die over februari is *verdiend*? Stel dat er in juni een nieuwe CAO ingaat en de persoon er met terugwerkende kracht vanaf januari maandelijks 50 euro bij krijgt. Corrigeert de werkgever dan de aangiften over januari t/m mei of geeft hij gewoon over juni 2.300 euro aan? Doet hij het eerste dan is hij een *Loon Over* werkgever en doet hij het laatste dan is hij *Loon In* werkgever. Een *Loon Over* werkgever onderkent in zijn administratie netjes de twee tijdsdimensies waarop de PA is gebaseerd, een *Loon In* werkgever werkt met één tijdsdimensie. Het vervelende is nu dat we in de PA beide soorten loon door elkaar hebben lopen en we niet kunnen zien met welk loonbegrip we te maken hebben. In het algemeen kun je zeggen dat grote bedrijven werken met *Loon Over* (twee tijdsdimensies) en het MKB met *Loon In* (één tijdsdimensie). Een goed geautomatiseerde *Loon Over* werkgever kan in beginsel aangifte doen op basis van *Loon In*, maar voor een *Loon In* werkgever en zijn salarisverwerker is het niet bepaald simpel om eventjes een tijdsdimensie aan de administratie toe te voegen. Joe the Plumber weet niets van wat een accountant *matching* noemt. Dus is gekozen voor het verplicht aangeven van loonbedragen op basis van een *Loon In* systematiek. Werkt u bij een groot bedrijf dan gaat u dat ook merken in een afname van het aantal correcties op uw loonopgaven, want uw loonbriefje moet natuurlijk gelijk lopen met de PA.

Vanuit de PA bekijken we als techneuten deze schijnbaar ingrijpende systeemwijziging glimlachend, net als *Loon Over* werkgevers met een goed opgezet salarispakket. Yes we can! Correcties met terugwerkende kracht blijven ook in een *Loon In* wereld bestaan, dus voor het in dit artikel besproken gegevensmodel maakt het geen verschil. En voor de omvang van de PA boeit het ook al niet echt. Dit is daarom de boodschap die we iedere ontwerper willen meegeven: is tijd in uw systeem belangrijk, maak dan uw systeem net als de PA tot een *total recall* product en kijk glimlachend toe hoe de wereld worstelt met tijdlijnen. Hoe groot de voordelen van een perfect geheugen zijn zullen we in de komende artikelen duidelijk maken. Het is altijd fijn om je tijd vooruit te zijn.

In de volgende aflevering van de Polis Papers bespreken we als onderwerp: het geautomatiseerd managen van opmerkelijke, onvolledige en corrupte gegevens.

René Veldwijk en Frido van Orden zijn partner bij FAA Partners, onderdeel van de Ockham Groep.