|  |
| --- |
| Aan |
| Alex van den Berg  Richard Hogenberg Paul Kuijk  Chris Povee  Henk Tent Becking  Bianca Verhage  Paul Water |
| Onderwerp |
| IBM Optim UWV implementatie |
|  |
|  |
|  |

***Aanleiding, doel.***

Om te kunnen voldoen aan de AVG, wordt door de DataFabriek gebruik gemaakt van het product Optim. Hiermee worden persoons-identificerende gegevens gemaskeerd.   
Dat maakt het mogelijk dat de DataFabriek gegevens kan leveren waarmee wel bepaalde patronen kunnen worden herkend maar waarmee het (bijna) onmogelijk wordt om specifieke personen te identificeren.

IBM, de leverancier van Optim, garandeert dat Optim in alle toekomstige versies, voor één on-gemaskeerde waarde precies één unieke gemaskeerde waarde levert.

Zouden er ooit dubbele gemaskeerde waardes voorkomen dan zou dat desastreuse gevolgen hebben voor de data consistentie van de DataFabriek. Er zouden verkeerde relaties kunnen worden gelegd tussen personen en uitkeringen, betalingen. Met als gevolg verkeerde conclusies en beslissingen met verregaande maatschappelijke impact.

Er is een discussie ontstaan tussen het team Lithium en solution architecten in hoeverre men als UWV dan nog mitigerende maatregelen zou moeten treffen en welke dat dan moeten zijn.

In deze memo worden de standpunten van beide teams uitgelegd. Een vorige versie van deze memo is door beide team beoordeeld en na aanpassingen door beide teams geaccordeerd.

Het advies zal nu in het reguliere voortbrengingsproces worden opgenomen.

***Technische implementatie***

IBM, de leverancier van Optim, garandeert dat Optim in alle toekomstige versies, voor één on-gemaskeerde waarde precies één unieke gemaskeerde waarde levert. Mits dezelfde saltkey wordt gebruikt. Een saltkey wordt toegevoegd in het algoritme om er voor te zorgen dat het Optim gebruik door UWV uniek is in de wereld. Zonder deze toevoeging (zout) is het ook voor andere Optim gebruikers niet mogelijk de UWV manier van maskering uit te voeren.

Het mag duidelijk zijn dat deze saltkey in een speciale plaats opgeslagen dient te worden, met een strikte toegangsregeling. Er is bepaald dat in elke omgeving (OTAP) een eigen saltkey wordt gebruikt.

Om het aantal aanroepen naar Optim te verminderen is besloten een maskeringstabel te gebruiken met daarin de on-gemaskeerde en gemaskeerde waardes.   
Door eerst deze tabel te raadplegen of voor een on-gemaskeerde waarde al eens een gemaskeerde waarde door Optim is bepaald, kan het aantal keren dat Optim aangeroepen moet worden drastisch worden beperkt.

De discussie gaat nu over de technische implementatie van het laadproces en de wijze waarop deze tabel wordt gebruikt om te waarborgen dat in deze tabel precies één on-gemaskeerde waarde met zijn onafscheidelijke metgezel één gemaskeerde waarde voorkomt.

De saltkey wordt opgeslagen IN de betreffende omgeving (OTAP) en is alleen toegankelijk voor Optim en personen met een speciale autorisatie. Door deze maatregel wordt voorkomen dat iemand die toegang heeft tot verschillende omgevingen gemaskeerde gegevens zouden kunnen herleiden tot individuele personen.

***Standpunt Lithium.***

Op de maskeringstabel is één index (IX) gedefinieerd die zowel de rubriek on-gemaskeerd als ook de rubriek gemaskeerd bevat en controleert dat er geen dubbele waardes (combinatie van on-gemaskeerd en gemaskeerd) worden geaccepteerd.

Het vullen van de maskeringstabellen is nu ondergebracht in een voorloopproces, dat is gepositioneerd nadat de data van een levering is ingelezen in DIM en voordat de Bronzone daarmee wordt bijgewerkt.   
In dit proces wordt IX gebruikt om te controleren of de waarde on-gemaskeerd al voorkomt in de tabel. Technisch: Dat is het eerste gedeelte van de index.   
Als de on-gemaskeerde waarde voorkomt, stopt het proces voor deze waarde en wordt de volgende verwerkt.   
Als de on-gemaskeerde waarde niet voorkomt, wordt Optim aangeroepen en de gemaskeerde waarde bepaald. In de tabel wordt een nieuw record gecreëerd met daarin de combinatie on-gemaskeerd en gemaskeerde waarde, Het DBMS zal daarbij ook de index (IX) bijwerken.

Om een extra slot op de deur te hebben, zou als laatste stap in het voorloop proces een extra CHECK kunnen worden ingebouwd die controleert of er dubbele on-gemaskeerde en /of gemaskeerde waardes voorkomen. Technisch is dit een tabel scan maar dat is niet erg omdat dit maar een keer gebeurd per voorloop proces.

***Standpunt Solution architecten.***

Er wordt niet getwijfeld aan de juiste werking van de door Lithium voorgestelde implementatie.

Deze is gebaseerd op de voorwaarde dat IBM de afgegeven garantie te allen tijde WAAR kan maken.

*IBM, de leverancier van Optim, garandeert dat Optim in alle toekomstige versies, voor één on-gemaskeerde waarde precies één unieke gemaskeerde waarde levert. Mits dezelfde saltkey wordt gebruikt.*

Contractueel is dit vastgelegd en als er dan toch dubbele waardes zouden voorkomen in productie dan valt UWV niets te verwijten. Maar er kan meer mis gaan dan alleen de correcte werking van Optim.  
  
Er kunnen menselijke fouten worden gemaakt door beheerders en/of ontwikkelaars.  
Er hoeft maar ergens een instelling te zijn aangepast waardoor Optim verkeerde (dubbele) resultaten zou kunnen leveren.

Voorbeelden:

* Verkeerde omgevingsinstellingen (Verwijzing naar saltkey).
* Bug in aanroep van Optim.
* Bug in laad programmatuur.
* Verkeerde installatie van nieuwe Optim versie.

Om te voorkomen dat UWV in een onherstelbare situatie terecht komt, wordt geadviseerd extra maatregelen te treffen. Het advies is om twee extra indexen te definiëren die uniciteit garanderen op de on-gemaskeerde rubriek en op de gemaskeerde rubriek.  
Daarmee wordt gewaarborgd dat het onmogelijk is om dubbele waardes in deze rubrieken te plaatsen en is de maskeringstabel altijd consistent en een juiste basis voor het verdere proces.

***Vergelijking.***

Het vullen van de maskeringstabellen in een voorloopproces en Optim alleen voor nieuwe waardes aanroepen, is een uitstekende oplossing om de aantal aanroepen naar Optim te verminderen en zo de performance te verbeteren. Het gebruik van IX zal ook bijdragen aan een betere performance en er wordt ook niet getwijfeld aan de juiste werking van deze implementatie.

Is het gebruik van IX samen met het extra slot op de deur CHECK, voldoende om te waarborgen dat deze situaties niet worden toegelaten in het systeem?

Is het toevoegen van de door solution architectuur extra indexen noodzakelijk en zijn deze de investering waard?

**Huidige opzet door Lithium.**  
Eerst wordt met IX gecontroleerd of een on-gemaskeerde waarde al voorkomt.  
Als de on-gemaskeerde voorkomt zal de Optim aanroep niet plaats vinden, dus is in het systeem maar één gemaskeerde waarde aanwezig.   
Hoewel Optim misschien een andere waarde zou kunnen opleveren wordt de nu bekende waarde gebruikt in het systeem. Voor deze situatie voldoet de huidige opzet.   
  
Als IX constateert dat de on-gemaskeerde waarde niet aanwezig is, wordt Optim aangeroepen en levert deze een gemaskeerde waarde op.  
Deze gemaskeerde waarde zou al kunnen voorkomen in de maskeringstabel, bij een andere on-gemaskeerde waarde en zou eigenlijk geweigerd moeten worden.   
Het voorloop proces zal deze toch toevoegen omdat voor IX de combinatie on-gemaskeerd en gemaskeerd een unieke combinatie bevat.  
Deze dubbele gemaskeerde waarde zal wel worden gesignaleerd door CHECK en zal weer verwijderd moeten worden. CHECK zal daarvoor alle waardes uit de tabel moeten lezen (table scan). Dat heeft enige performance impact maar aangezien dit maar een keer per voorloop proces gebeurd, lijkt dit acceptabel.  
  
Maar stel dat CHECK een dubbel vindt, wat dan?

De te verwijderen gemaskeerde waarde staat dan meerdere malen in de maskeringstabel en verwijst naar meerdere on-gemaskeerde waardes. Welke moet worden verwijderd en wat gebeurt er dan met de on-gemaskeerde waarde die dus nu GEEN gemaskeerde waarde krijgt?

De belangrijkste stap zou zijn om zoveel mogelijk informatie te rapporteren zodat een oorzaak kan worden gevonden.

* Welk “Voorloop proces” heeft dit veroorzaakt?
* Uit welke aanlevering is dit afkomstig?
* Welk record in deze aanlevering?
* Wat was de on-gemaskeerde, gemaskeerde waarde.
* Welke aanroep naar Optim is gedaan en met alle details (invoer, uitvoer, versie, tijdstip etc.).

CHECK zal niet voldoende informatie kunnen afleiden uit alleen de maskeringstabel maar ergens extra informatie vandaan moeten halen. Dat lijkt een onmogelijke taak.

De vraag is gesteld of deze opzet ook werkt indien er meerdere voorloop processen tegelijk actief zijn?

De nu ontwikkelde programmatuur moet verticaal schaalbaar zijn om de toekomstige werklast aan te kunnen. Verdelen van werklast kan door meerdere aanleveringen tegelijk te verwerken of zelfs door grote aanleveringen op te splitsen in kleinere stukjes die elk door een eigen proces instantie worden verwerkt. Van een proces kunnen best meerdere instanties tegelijk actief zijn.  
  
Als er meerdere voorloop processen tegelijk kunnen draaien, is het mogelijk dat het eerste proces constateert dat de on-gemaskeerde waarde niet voorkomt en dan Optim gaat aanroepen.   
Een tweede proces kan gedurende deze aanroep tot dezelfde conclusie komen en ook Optim aanroepen. Het is echter onwaarschijnlijk dat Optim twee verschillende waardes zal leveren.   
Immers er is niets veranderd in de instellingen in de afgelopen paar seconden en de garantie van IBM lijkt dan toch voldoende.

IX bewaakt bovendien dat de combinatie (on-gemaskeerd, gemaskeerd) maar 1 keer mag voorkomen. Het tweede proces kan dan ook een “bestaat al” melding van IX negeren.

Voor zover voldoet de huidige opzet.   
CHECK zal bovendien achteraf nog eens controleren of er dubbele gemaskeerde waardes voorkomen en deze indien nodig verwijderen.   
Zoals beschreven heeft CHECK extra informatie nodig om een nuttige rapportage te kunnen maken. Door de meerdere proces instanties zal het vergaren van die informatie extra worden bemoeilijkt.

Conclusie: De gekozen opzet werkt ook in geval van meerdere actieve voorloop processen.

**Voorgestelde opzet door Solution architecten.**

Door gebruik te maken van het DBMS is er zekerheid dat een foutieve situatie niet wordt toegelaten in de database. De voorgestelde extra index op on-gemaskeerd en gemaskeerd staat niet toe dat er dubbele waardes voorkomen.

Het CHECK proces is dan overbodig omdat het voorloop proces al tijdens het registreren door het DBMS (Oracle) wordt gewezen op een foutieve situatie. Dan kan het voorloop proces meteen actie ondernemen.  
Bovendien beschikt het voorloop proces over alle details om hier over te rapporteren. (Welke bron, aanroep Optim etc.).

***Advies.***

Het advies is om niet ALLEEN te vertrouwen op de door IBM afgegeven garantie omdat er door andere factoren, zoals een menselijke (installatie) fout, dubbele gemaskeerde waardes zouden kunnen voorkomen.

Dat zou dan desastreuse gevolgen hebben voor de data consistentie van de DataFabriek. Er zouden verkeerde relaties kunnen worden gelegd tussen personen en uitkeringen, betalingen. Met als gevolg verkeerde conclusies en beslissingen met verregaande maatschappelijke impact.

De huidige opzet verminderd dit risico aanzienlijk door na elk voorloop proces een extra controle (CHECK) te verrichten op dubbele on-gemaskeerde en/of gemaskeerde waardes maar dit CHECK proces beschikt dan niet meer over voldoende informatie om in detail te kunnen rapporteren. Dat bemoeilijkt de analyse van het probleem.

De voorgestelde oplossing om een extra index te definiëren op gemaskeerd, zorgt ervoor dat het onmogelijk is dubbele waardes te registreren. Een voorloop proces “wordt meteen op de vingers getikt” en beschikt over voldoende informatie om in detail hierover te rapporteren.

Lithium geeft aan dat het vergaren van extra informatie ook in dit voorloop proces moeilijk zal zijn.   
Er rest ook dan niets anders dan het proces te beëindigen en te signaleren dat er dubbele waardes zijn gevonden. Daarbij is alleen het leveren van de dubbele on-gemaskeerde en gemaskeerde waardes mogelijk. Deze extra index levert weliswaar een lichte performance en opslag penalty maar de winst is dan weer dat het CHECK proces overbodig is.

De voorgestelde index op on-gemaskeerd is niet nodig omdat in het proces eerst wordt gecontroleerd of on-gemaskeerd voorkomt in de maskeringstabel alvorens Optim aan te roepen.   
Daarmee garandeert het proces er geen dubbele on-gemaskeerde in de maskeringstabel terecht komen.  
Er is wel een kleine kans aanwezig als meerdere voorloop processen tegelijk actief zouden kunnen zijn.  
Maar die kans is verwaarloosbaar klein.

Zou de huidige IX (combinatie on-gemaskeerd en gemaskeerd) vervangen moeten worden door een index op alleen on-gemaskeerd?

Het mag duidelijk zijn dat veel processen de maskeringstabel zullen raadplegen met als invoer on-gemaskeerd en gewenste uitvoer gemaskeerd.  
De huidige IX ondersteund deze vertaling ZONDER de feitelijke database tabel te hoeven raadplegen.

De index zelf bevat voldoende informatie om dit te ondersteunen.  
Per definitie levert dit een betere performance voor al deze raadpleeg functie op.  
Advies is de huidige IX te gebruiken waarmee de raadpleeg functie uitstekend en snel kan worden ondersteund.

Tijdens de review van deze memo kwam Lithium nog met een voorstel om te controleren of er menselijke (installatie) fouten zijn gemaakt.

Dat kan eenvoudig door een aantal waardes te maskeren met Optim en te bewaren als controle set.

Voor elke run kunnen kan deze controle set worden gebruiken om te controleren of Optim nog het zelfde resultaat levert. Daarmee wordt zeker gesteld dat alle omgevingsfactoren nog correct zijn.

Ook bij het installeren van een nieuwe versie van Optim zal dit een waardevolle actie zijn.

Deze memo neemt dit voorstel graag als advies over.

Samengevat advies:

* Definieer een nieuwe unieke index op gemaskeerde waardes.
* Breek het voorloop proces af als er dubbele waardes worden gesignaleerd door de nieuwe index.  
  Vergaar zoveel mogelijk informatie ter ondersteuning van het onderzoek naar de oorzaak.
* Het CHECK proces is overbodig.
* Gebruik een controle set om te controleren of de maskering nog op dezelfde wijze gebeurd.