Performanceproblemen zijn met goede keuzes te voorkomen

# (Basis)administraties en gegevensuitwisseling

René van Leusen

**In de praktijk blijkt regelmatig dat het uitwisselen van gegevens tussen systemen zeer ingewikkeld is. Het gaat dan bijvoorbeeld om complexe berichten en ingewikkelde afspraken om deze berichten samen te stellen en te verwerken. Maar ook om het niet kunnen leveren van gegevens, omdat bij het ontwerpen van een administratie geen rekening is gehouden met bepaalde informatiebehoeftes van de afnemende systemen.**

Bij omvangrijke en grote aantallen berichten is in productie vaak sprake van performanceproblemen en gebrek aan hardware resources (CPU, geheugen). Voor de korte termijn zijn het toepassen van noodgrepen in de code en het aanschaffen van dure hardware meestal de enige oplossingsalternatieven. Aanpassen van afgestemde berichten en vastgelegde afspraken is een langdurend traject.
Het uitwisselen van gegevens tussen systemen wordt steeds belangrijker. Het centraal en eenmalig bewaren van veel gebruikte gegevens is een architectuurprincipe dat veel wordt toegepast. Een bekend voorbeeld betreft de basisadministraties die de overheid heeft onderkend (nl.wikipedia.org/wiki/Basisregistratie). Een ander voorbeeld is masterdata management waarbij organisaties veel gebruikte gegevens zoals bijvoorbeeld referentietabellen en klantgegevens centraal bewaren (en.wikipedia.org/wiki/Master_Data_Management). In dit artikel wordt een aantal onderwerpen beschreven die tijdens het ontwerpen van een (basis)administratie en het uitwisselen van gegevens met andere systemen meegenomen moeten worden. De gemaakte keuzes bepalen in hoge mate de functionaliteit, complexiteit, onderhoudbaarheid, performance en resourcegebruik van de systemen. Het gaat om de volgende onderwerpen:

1. Systemen hebben geen kennis van de business logica van andere systemen;
2. Uitwisselen van (gewijzigde) gegevens tussen systemen;
3. Uitwisselen van (gewijzigde) gegevens in een keten van systemen;
4. Uitwisselen van (gewijzigde) gegevens afhankelijk van business rules;
5. Uitwisselen van gegevens (on)afhankelijk van de databasestructuur;
6. Uitwisselen van gegevens afhankelijk van de gewenste integriteit;
7. Mutatiehistorie van gegevens;
8. Geldigheid van gegevens.

De eerste vijf onderwerpen gaan over het uitwisselen van gegevens tussen systemen. De laatste drie onderwerpen gaan over welke eisen er gesteld worden aan de gegevens(structuur) van de lokale database van een systeem, opdat een systeem de juiste gegevens kan uitwisselen met andere systemen.

## Voorbeeld deelauto

Bovenstaande onderwerpen worden toegelicht aan de hand van een voorbeeld over het gezamenlijk gebruik van een aantal deelauto's door een aantal huisadressen. Op een huisadres wonen een of meerdere bestuurders. De algehele administratie van het deelautoproject wordt vanuit één centraal huisadres gecoördineerd.
In de uitwerking van bovenstaande onderwerpen wordt er vanuit gegaan dat het centrale huisadres, alle huisadressen en alle bestuurders een eigen systeem hebben voor het uitwisselen van gegevens over rekeningen en roosters. Het uitwisselen van rekeninggegevens (roostergegevens is identiek) tussen deze systemen ziet er uit als in afbeelding 1. Verder hebben alle systemen een eigen database voor het opslaan van gegevens. De gegevensstructuur van de database van het centrale huisadres ziet er uit zoals in afbeelding 2.
In RIT wordt bijgehouden welke BESTUURDER wanneer hoeveel kilometer heeft gereden in welke AUTO. In BOETE wordt bijgehouden tijdens welke RIT een overtreding is beboet, en in ROOSTERUUR kan een HUISADRES een auto reserveren.

**1. Systemen hebben geen kennis van de business logica van andere systemen.**
Aan het einde van elke maand stuurt het centrale huisadres een rekening naar alle deelnemende huisadressen met het verzoek om de rekening te betalen. De huisadressen sturen daarop een rekening naar de op dat adres wonende bestuurders met het verzoek om elk hun deel te betalen. Een interessante vraag is wie het beste het bedrag kan berekenen dat elke bestuurder moet betalen: het centrale huisadres of de verschillende huisadressen? Als wordt afgesproken dat het centrale huisadres dit bedrag moet berekenen, dan moet het centrale huisadres weten op welke

manier de verschillende huisadressen de rekening van het huisadres verdelen over de verschillende bestuurders die wonen op dat huisadres. Het is immers mogelijk dat bij het ene huisadres de bestuurders betalen naar gebruik, en bij het andere huisadres naar draagkracht. In de meeste complexe situatie heeft elk huisadres een andere verdeelsleutel. Een belangrijk nadeel is de sterke afhankelijkheid: als een huisadres besluit om de eigen verdeelsleutel te wijzigen, dan moet de aanpassing bij het centrale huisadres gebeuren. De beste keuze is om de verschillende huisadressen het bedrag zelf te laten berekenen. Bij deze berekening kan een huisadres zijn eigen verdeelsleutel gebruiken. Het centrale huisadres stuurt slechts gegevens en hoeft geen kennis te hebben van de verdeelsleutels van de verschillende huisadressen.

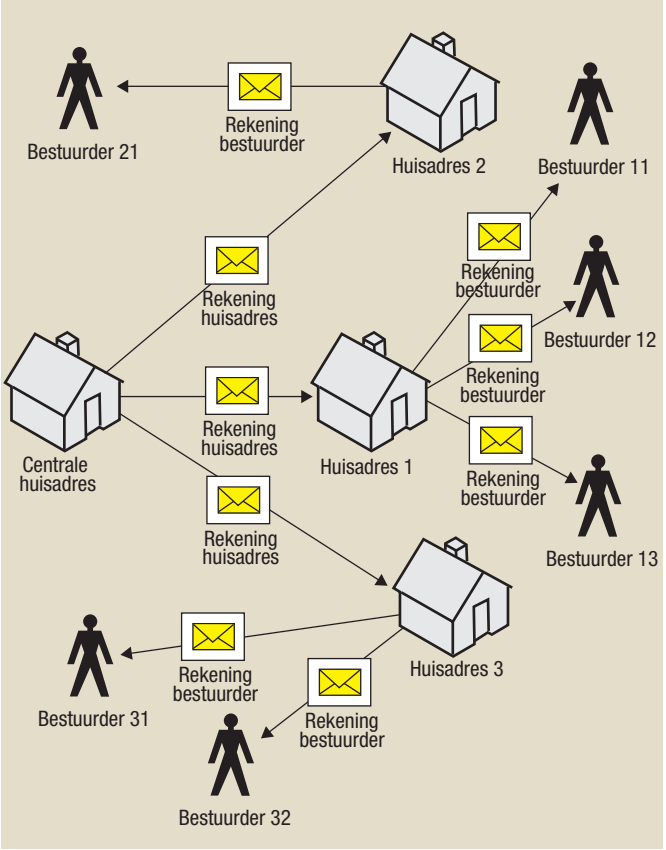
**2. Uitwisselen van (gewijzigde) gegevens tussen systemen.**
Op de laatste dag van de maand stuurt het centrale huisadres een rooster naar alle deelnemende huisadressen van welk huisadres welke auto de komende maand wanneer tot zijn beschikking heeft. Om optimaal gebruik te kunnen maken van de verschillende (deel)auto's is afgesproken dat elke bestuurder mutaties door moet geven aan het centrale huisadres. Het centrale huisadres geeft alle mutaties aan het eind van de dag eenmalig door aan alle huisadressen. Het doorgeven van mutaties kan via meerdere methodes. Hier zijn twee veel voorkomende methodes:

- Het centrale huisadres stuurt een nieuw rooster waarin de mutaties zijn verwerkt;
- Het centrale huisadres stuurt alleen mutaties.

De eerste methode, het centrale huisadres stuurt een nieuw rooster waarin de mutaties zijn verwerkt, is voor het centrale huisadres de gemakkelijkste methode. Het centrale huisadres hoeft niet bij te houden welke wijzigingen er zijn opgetreden sinds het versturen van het laatste rooster en kan op elke moment opnieuw het rooster samenstellen en versturen. Voor de ontvangende huisadressen is deze methode een stuk lastiger. Het is immers niet direct duidelijk wat er is gewijzigd ten opzichte van het vorige rooster. Vergelijken van het oude en nieuwe rooster kan zeer complex zijn, en het lezen van het oude rooster uit de lokale huisadresdatabase kost tijd en resources.
De tweede methode, het centrale huisadres stuurt alleen de mutaties, is voor het centrale huisadres is veel lastiger: het centrale huisadres moet bijhouden welke mutaties er zijn gebeurd sinds het versturen van het laatste rooster. Het succes van deze methode is afhankelijk van het correct verwerken van alle mutaties. In theorie zou dit geen problemen mogen opleveren, echter in de praktijk blijkt regelmatig dat administraties na verloop van tijd, ondanks alle goede afspraken, toch uit elkaar gaan lopen. Voor het ontvangende huisadres is deze methode gemakkelijker, het is immers direct duidelijk wat de mutaties zijn.

**3. Uitwisselen van (gewijzigde) gegevens in een keten van systemen**.
Dit onderwerp is een uitbreiding van het voorgaande onderwerp. Voor een huisadres is het ongewijzigd doorsturen van alle ontvangen berichten van het centrale huisadres naar de bestuurders de gemakkelijkste oplossing. Lastiger is de situatie als een huisadres bij een roosterwijziging *het complete rooster* ontvangt van het centrale huisadres, terwijl de bestuurders *alleen de roosterwijzigingen* willen ontvangen. Het huisadres moet in dat geval de roosterwijzigingen bepalen door het vergelijken van het nieuwe rooster met het vorige rooster. Vergelijken is complex, betekent veel database acties, en kost veel tijd en resources.

Bij het bepalen van de roosterwijzigingen moet het huisadres een keuze maken hoe sterk de koppeling mag zijn tussen het verwerken van het inkomende bericht van het centrale huisadres en het samenstellen van het uitgaande bericht voor de verschillende bestuurders. Een veel gebruikt architectuurprincipe zegt dat de database als ontkoppelpunt moet fungeren. Voordeel van dit principe is dat verwerken van inkomende berichten onafhankelijk is van het samenstellen en versturen van uitgaande berichten. Nadeel van dit principe is vaak een slechtere performance en meer resourcegebruik. Toelichting: wijzigingen die bepaald zijn bij het verwerken van het inkomende bericht van het centrale huisadres zouden én gebruikt kunnen worden om de lokale database van het huisadres aan te passen, én gebruikt kunnen worden om de uitgaande berichten voor de bestuurders samen te stellen. Voordeel van deze oplossing is dat de wijzigingen maar 1 keer bepaald hoeven te worden. Nadeel is de sterke koppeling



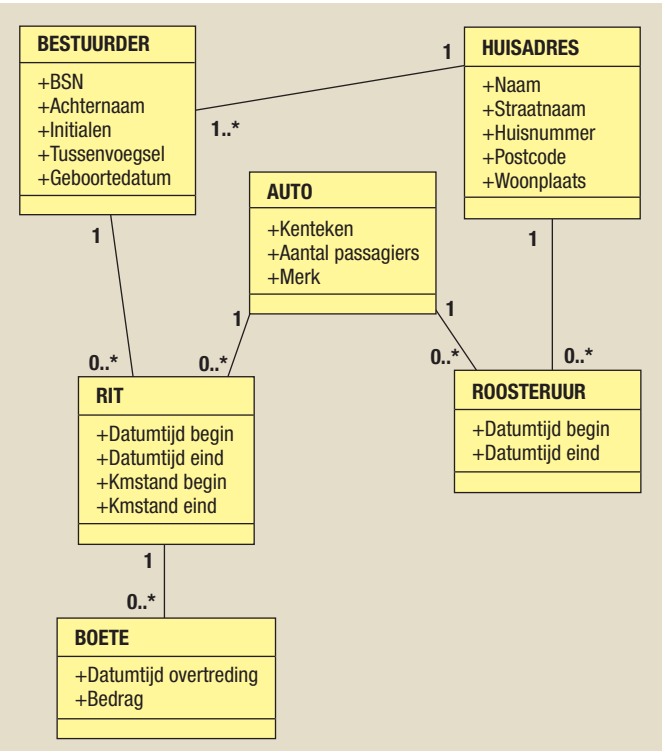

**Afbeelding 1**: Alle huisadressen en bestuurders hebben een eigen systeem voor uitwisseling van gegevens.

tussen verwerken van het inkomende bericht van het centrale huisadres en het samenstellen van de uitgaande berichten voor de bestuurders. Als het inkomende bericht van het centrale huisadres wijzigt, dan kan dit impact hebben op het samenstellen van de uitgaande berichten voor de bestuurders. Een kleine, op het eerste gezicht snelle en goedkope functionele wijziging kan door deze sterke koppeling een grote, langdurende en kostbare technische aanpassing betekenen. Dit is onwenselijk. Een andere mogelijkheid (conform het bovenstaande architectuurprincipe) is om bij het samenstellen van de uitgaande berichten voor de bestuurders de wijzigingen opnieuw te bepalen uitgaande van alleen de databasegegevens. Nadeel zijn een slechtere performance en meer resourcegebruik.

**4. Uitwisselen van (gewijzigde) gegevens afhankelijk van business rules.**

Niet alle (gewijzigde) gegevens mogen en hoeven verstuurd te worden naar alle afnemende systemen. Vanuit privacy-overwegingen is het niet wenselijk dat de bestuurders van elkaar weten hoeveel boetes de ander moet betalen. Tussen het centrale huisadres, de huisadressen en de bestuurders moet afgesproken worden onder welke condities (gewijzigde) gegevens verstuurd mogen en moeten worden. Voor de rekening van de bestuurder geldt dat deze alleen boetes mag bevatten die de bestuurder zelf moet betalen.

Een ander voorbeeld is als een roosterwijziging dezelfde dag door een andere roosterwijziging wordt teruggedraaid. Stel; een bestuurder reserveert 's ochtends een auto voor de volgende dag, wordt 's middags gebeld voor een afspraak voor de volgende dag




**Afbeelding 2**: Gegevensstructuur database centrale huisadres.

en besluit daarop de reservering voor de volgende dag te annuleren. Moet in dit geval het centrale huisadres aan het einde van de dag een nieuw rooster versturen? Voor de ontvangende huisadressen is er immers niets gewijzigd. Als afgesproken wordt om in dit geval geen rooster te versturen dan moet het centrale huisadres kijken of wijzigingen elkaar opheffen. Het is dan niet voldoende om te kijken of er records in de centrale database zijn met een later registratietijdstip dan het tijdstip van versturen van het laatste rooster naar de verschillende huisadressen.

**5. Uitwisselen van gegevens (on)afhankelijk van de databasestructuur.**

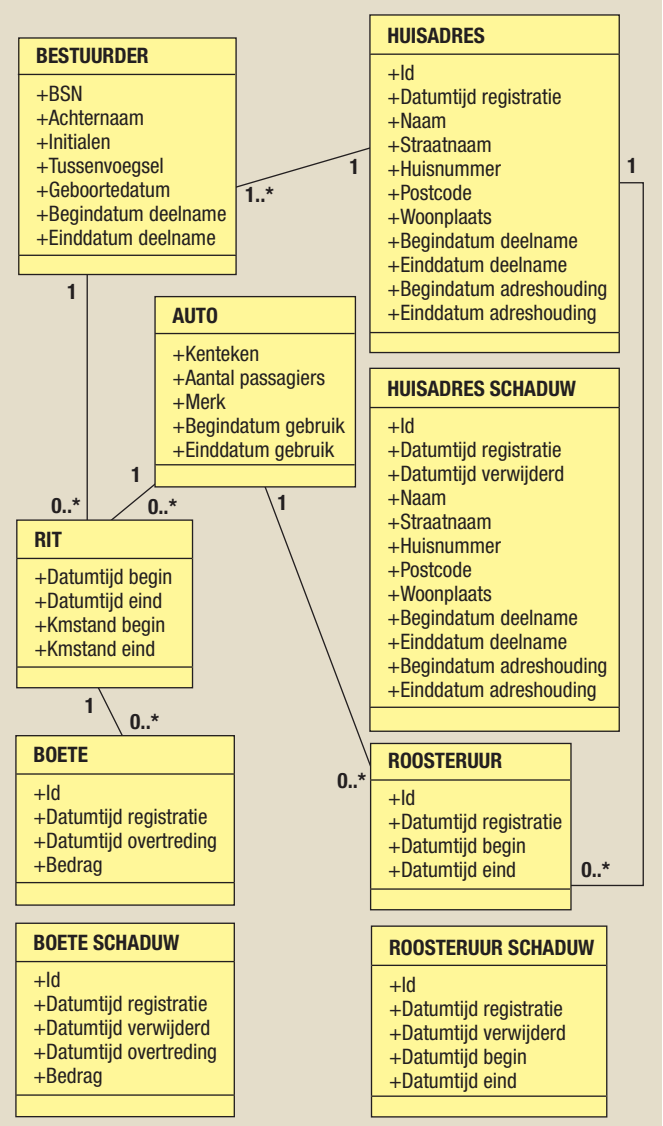
Een veel gebruikt architectuurprincipe is dat de structuur van een bericht onafhankelijk moet zijn van de structuur van de database waarin de gegevens zijn geregistreerd. Dit geldt zowel voor het systeem dat het bericht samenstelt en verstuurt, als het systeem dat het bericht ontvangt en verwerkt. Het idee is dat als de databasestructuur van een systeem wijzigt, dit geen gevolgen heeft voor de structuur van het bericht en zeker geen gevolgen voor de databasestructuur van het andere systeem. Eventuele structuurwijzigingen moeten opgevangen worden in het samenstellen en verwerken van de berichten.

Bij het samenstellen en verwerken van het bericht is de databasestructuur wel belangrijk. Bij het samenstellen en verwerken van een XML bericht moet een vertaling gemaakt worden tussen een relationele databasestructuur en een hiërarchische XML structuur. Een hiërarchische structuur is een speciale relationele structuur. Namelijk elke entiteit heeft maximaal 1 ouder. Om van een relationele structuur een hiërarchische structuur te maken, moet van elke entiteit in de relationele structuur die onderdeel uit gaat maken van het bericht, het aantal ouderrelaties teruggebracht worden tot maximaal 1 ouderrelatie. De complexiteit en performance van deze vertaling wordt onder andere bepaald door de mate waarin de hiërarchische XML structuur en de relationele databasestructuur op elkaar aansluiten.

De bovenstaande vijf onderwerpen gingen over het uitwisselen van gegevens tussen systemen. De laatste drie onderwerpen gaan over welke eisen er gesteld worden aan de gegevens-(structuur) van de database, opdat een systeem de juiste gegevens kan uitwisselen met andere systemen. In de onderstaande drie onderwerpen wordt verwezen naar de gegevensstructuur van afbeelding 3.

**6. Uitwisselen van gegevens afhankelijk van de gewenste integriteit.**

Wat te doen als de ontvanger van een bericht de integriteit van de verstuurde gegevens wil controleren? Met integriteit wordt bedoeld dat de gegevens niet zijn gewijzigd na vastlegging. In het deelautoproject wil een bestuurder kunnen controleren dat de ritgegevens niet zijn gewijzigd sinds het moment van vastleggen in de administratie van het centrale huisadres. Deze ritgegevens worden immers gebruikt bij het bepalen van de rekening. Achteraf controleren kan door bij het vastleggen van de ritgege-




**Afbeelding 3**: Uitgebreide gegevensstructuur.

vens ook direct een digitale handtekening van de ritgegevens vast te leggen. Wel moeten alle ritgegevens die gebruikt zijn bij het berekenen van de digitale handtekening opgenomen worden in de rekening, omdat anders de bestuurder niet in staat is om de integriteit van de ritgegevens te controleren. In de gegevensstructuur van afbeelding 3 is RIT uitgebreid met een digitale handtekening.

**7. Mutatiehistorie van gegevens.**

Als een bestuurder het niet eens is met een boete heeft hij of zij het recht om beroep aan te tekenen. Wat nu als het beroep wordt gehonoreerd en de boete wordt ingetrokken? Als de boete fysiek verwijderd wordt uit de database van het centrale huisadres, dan kan achteraf de vraag van een bestuurder waarom een (reeds verstuurde) rekening een (ingetrokken) boete bevat, niet beantwoord worden. Als de boete logisch wordt verwijderd en het tijdstip van verwijderen wordt vastgelegd, dan kan later voor elk moment in het verleden bepaald worden welke boetes voor welke bestuurder bekend waren in de database van het centrale huisadres. In de bovenstaande gegevensstructuur is voor BOETE, ROOSTERUUR en HUISADRES een schaduwtabel opgenomen met daarin een aantal extra attributen. Versies van hetzelfde gegeven hebben dezelfde ID, 'ID + Datumtijd registratie' is de unieke sleutel van een schaduwtabel, en 'Datumtijd verwijderd' gevuld betekent dat het record is verwijderd op dat tijdstip.

**8. Geldigheid van gegevens.**

Binnen het project deelauto worden auto's elke vier jaar vervangen. Stel; op de eerste dag van de nieuwe maand wordt een auto vervangen. Dit betekent dat deze auto niet meer gereserveerd kan worden voor gebruik na dit tijdstip. Om dit mogelijk te maken wordt AUTO voorzien van een geldigheidsperiode.
Tijdens het reserveren van een auto moet gecontroleerd worden of het roosteruur binnen de geldigheidsperiode valt.
Het project deelauto bevat adresgegevens. Veel systemen hebben een koppeling met de Gemeentelijke basisadministratie voor persoonsgegevens (GBA) voor het leveren van persoons- en adresmutaties. De GBA bevat het adresgegeven 'Datum aanvang adreshouding'. Dit gegeven geeft aan wanneer een persoon op een bepaald adres is gaan wonen en dit gegeven is meestal niet gelijk aan de datum waarop het bericht van de GBA wordt ontvangen en/of verwerkt. Bij een verhuizing is de nieuwe bewoner verplicht om zich binnen vijf dagen na verhuizing in te laten schrijven bij de gemeente. De gemeente stuurt dit gegeven door naar de GBA, waarna de GBA het weer doorstuurt naar alle afnemers van adresmutaties voor die persoon. 'Datum aanvang adreshouding' kan als 'Begindatum adreshouding' van een HUISADRES worden gebruikt.
In de gegevensstructuur van afbeelding 3 zijn BESTUURDER, AUTO en HUISADRES uitgebreid met een geldigheidsperiode.

## Conclusie

Als tijdens het ontwerpen van een (basis)administratie en het berichtenverkeer ten behoeve van het uitwisselen van gegevens met andere systemen rekening wordt gehouden met de onderwerpen die in dit artikel beschreven zijn, dan heeft dit grote voordelen voor de functionaliteit, complexiteit, onderhoudbaarheid, performance en het resourcegebruik van de systemen. Het oplossen van performanceproblemen en gebrek aan hardware resources in productie, door het toepassen van noodgrepen in de code en het aanschaffen van dure hardware, kunnen op deze manier worden voorkomen.

**Literatuur**

*Toon Loonen, Mutatiehistorie van gegevens, Database Magazine 3, 1999.*

**René van Leusen** (rene.van.leusen@capgemini.com) is als integratie-architect werkzaam bij Capgemini.

*Er bestaat een uitgebreidere versie van dit artikel, die u kunt vinden op www.dbm.nl onder het menu Speciaal > Extra materiaal.*