IN UITGELICHT brengt de redactie van Zorgvisie ict een ontwikkeling of een onderwerp extra onder de aandacht van de lezer.

Angst voor vreemde databases

Moeten **applicaties** wel of niet onafhankelijk zijn van de onderliggende database? Bart Groothuis en Jurrie Zaat zetten de voor- en nadelen van **databaseonafhankelijkheid** naast elkaar.

AUTEURS BART GROOTHUIS EN JURRIE ZAAT **BEELD** M&I PARTNERS

Zeker in een markt waarin standaardsoftwarepakketten gebruikelijk zijn, is het een terugkerend thema: databaseonafhankelijkheid. Bij de keuze voor een nieuwe standaardapplicatie staat vaak in het programma van eisen dat het systeem moet kunnen draaien op diverse databaseplatformen. Deze wens komt voort uit het streven naar standaardisatie. Als je immers al vier Oracle databases hebt, wil je dat het nieuwe pakket ook op Oracle draait en niet op Progress, om maar eens een voorbeeld te noemen.

In veel sectoren, zo ook in de IT, heerst de gedachte dat standaardisatie leidt tot kostenverlaging. In zijn algemeenheid is dat waar. De aanname bij standaardisatie op databaseplatform is dat dit leidt tot lagere beheerskosten en gunstiger licentietarieven, maar dat ligt wat genuanceerder. Het heeft alles te maken met de hoeveelheid beheer die verschillende databaseplatformen nodig hebben. Een database als Progress of MySQL vraagt nauwelijks beheer, terwijl voor Oracle de functie 'database administrator' (DBA) specifiek is uitgevonden.

Stel nu dat een organisatie met zo'n 2000 medewerkers vijftig medewerkers in de backoffice heeft die kleine applicaties gebruiken die op Oracle draaien. Voor deze kleine applicaties is het Oracle-beheer geen heel grote taak. In het primaire proces wordt een Progress-applicatie gebruikt door zo'n 1800 medewerkers. Vanuit het oogpunt van standaardisatie wil men deze primaire procesapplicatie nu ook op een Oracle databaseplatform gaan draaien. Dan blijkt plots dat het Oraclebeheer een serieuze aangelegenheid wordt.

DE AANNAME BIJ STANDAARDISATIE OP DATABASEPLATFORM IS DAT DIT LEIDT TOT LAGERE BEHEERSKOSTEN

Applicatiearchitecturen

Een klein uitstapje naar de architectuur van grote informatiesystemen. Vroeger, in de tijd van de COBOL-programma's, bestond een informatiesysteem uit diverse programma's. Ieder programma vormde een geïntegreerd geheel van gebruikersinterface, logica en data-afhandeling. De data werden opgeslagen in simpele bestanden zonder veel intelligentie. De logica en de gebruikersinterface waren één geheel. Een business rule, bijvoorbeeld dat een datum-in-dienst moet liggen ná de geboortedatum van de betreffende persoon, werd geprogrammeerd in de applicatie en deze regel werd in ieder (deel-)programma dat iets deed met de gegevens 'datum-in-dienst' en 'geboortedatum' apart opgenomen.

Ten tijde van de ontwikkeling van client-serverapplicaties werd de datalaag gescheiden van de gebruikersinterface en de logica. De data stonden op een server en de applicatie draaide op een pc/desktop. De term 2-tier deed zijn intrede.

Tegenwoordig bestaat een moderne toepassing, of dat nu een standaardapplicatie is of een maatwerkproduct, eigenlijk uit drie grote, veelal afzonderlijke componenten (zie de figuur op pagina 26). De zogenoemde 3-tier omvat een dataopslag, gebruikersinterface en een deel waar de logica (business rules) in wordt afgehandeld.

Omdat applicaties steeds minder op zichzelf staan maar functioneren in een netwerk van andere applicaties waarmee informatie wordt uitgewisseld, is het belangrijk dat de

business rules altijd worden toegepast. Als deze op de een of andere manier worden omzeild, komt de consistentie van de gegevens in de dataopslag in het geding. Daarom wordt als goede gewoonte bij softwareontwikkeling de gebruikersinterface gescheiden van de logicalaag. Op die manier is het bovendien makkelijker om bijvoorbeeld alleen de gebruikersinterface van een applicatie te vervangen of moderniseren, zonder dat de data-integriteit gevaar loopt en alle functionaliteit opnieuw moet worden getest. Bij het koppelen van een applicatie met andere systemen is het belangrijk dat altijd wordt aangesloten op de business rule-component in de logicalaag en niet rechtstreeks op de database. In de praktijk gebeurt dat echter regelmatig anders, bijvoorbeeld doordat een goede aansluitmogelijkheid bij een applicatie (ook wel API genoemd, Application Programming Interface) ontbreekt.

Ongebruikte functionaliteit

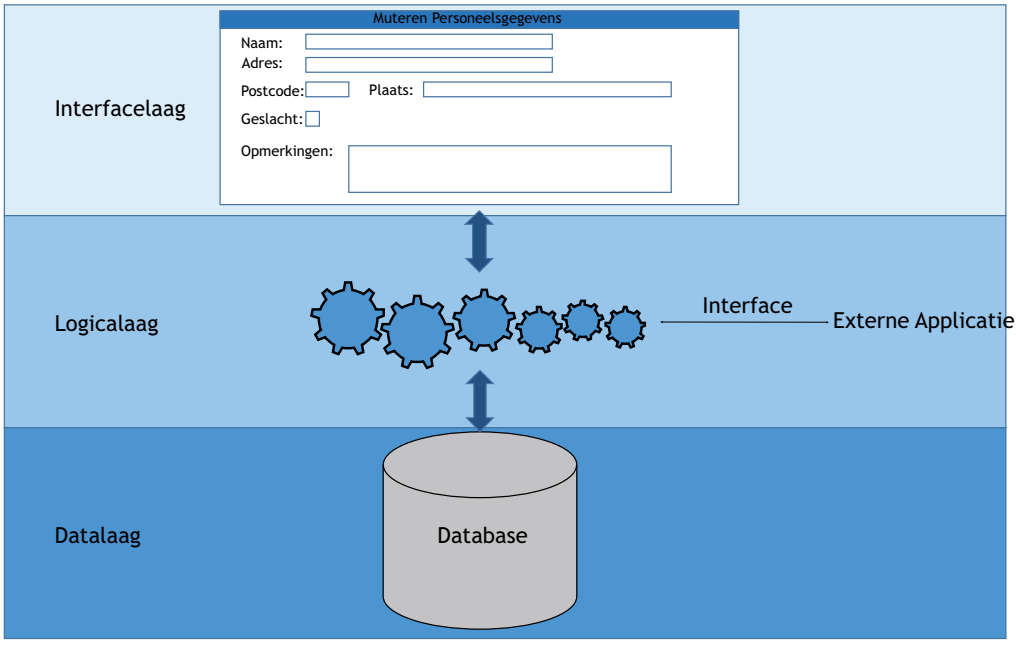
De afgelopen decennia hebben leveranciers van databases enorm veel functionaliteit toegevoegd aan hun producten. Databanken regelden in het verleden uitsluitend de opslag van gegevens in tabellen. In de loop van de tijd zijn ze geëvolueerd tot geavanceerde platforms met de introductie van autorisatielagen, integriteitscontroles, stored procedures, triggeringmechanismen en zelfs embedded fileservers, webservers en job >

> schedulers. De combinatie van databasetriggers (programma's die automatisch worden gestart bij het toevoegen, wijzigen en/of verwijderen van data in tabellen) en stored procedures (programmacode die in de database zelf wordt opgenomen) kan afdwingen dat business rules altijd worden toegepast, ongeacht via welke weg (gebruikersinterface, koppeling met ander programma) de data worden gemanipuleerd. Anders dan bij fysiek gescheiden lagen (data en business rules) kan dan wél rechtstreeks met de database worden gekoppeld, zonder dat de applicatielogica wordt buitengesloten.

In tegenstelling tot de standaardtaal SQL, die overigens vele dialecten kent, zijn al die extra opties leverancierspecifiek en daardoor niet zomaar uitwisselbaar. Softwareleveranciers van pakketten die meerdere SQL-databases ondersteunen, gebruiken daarom alleen die opties van een database die altijd aanwezig en universeel zijn: het verbinden met de database en de opslag van data. Dat is natuurlijk doodzonde: al die mooie opties die het moderne databaseplatform biedt worden niet gebruikt. En je betaalt er natuurlijk wel voor. Alleen al de instaplicentie van een database van een van de grote merken kost algauw duizenden euro's per jaar.

Stijgende pakketlicentiekosten

Om de softwareapplicatie meerdere databaseplatformen te kunnen laten ondersteunen, moet de leverancier flink investeren. Ten slotte moet hij minimaal testen of zijn applicatie ook echt werkt met al die verschillende SQL-databases. Dat betekent kosten voor de licenties van al die databaseplatformen, het inhuren of in dienst nemen van personeel met de benodigde kennis, testen, extra infrastructuur, et cetera. Uiteraard verrekent de softwareleverancier die kosten in zijn licentiemodel en berekent ze dus door aan de klant. Die klant kiest echter maar één optie, want die wilde immers standaardiseren op databaseplatformen, maar betaalt wel voor de ontwikkelkosten voor al die andere databaseplatformen. Kortom: de klant betaalt aanmerkelijk hogere licentiekosten voor functionaliteit die hij niet gebruikt.



AL DIE MOOIE OPTIES DIE HET MODERNE DATABASEPLATFORM BIEDT WORDEN NIET GEBRUIKT

Als de applicatieontwerper naar een applicatie streeft die databaseonafhankelijk is, dan kan deze natuurlijk niet de business rules opslaan in de database, hoewel alle grote merken daarvoor mogelijkheden hebben. Daarom wordt ervoor gekozen de business rules vast te leggen in de applicatie zelf en zelfs niet altijd als aparte component, maar gecombineerd met de gebruikersinterface. Wanneer de applicatie door middel van draaitabellen in Excel of door een interface met een ander pakket, de database direct benadert, omzeilt deze alle bedrijfslogica (business rules) met alle risico's van dien. Het zal duidelijk zijn wat onze conclusie is: streven naar applicaties die databaseplatform-onafhankelijk zijn, komt voort uit verouderd denken over databaseplatformen. Veel verstandiger is het te streven naar de standaarduitwisselbaarheid van gegevens. Iedere zorginstelling kent immers het fenomeen van vele maatwerkkoppelingen en interfaces tussen de diverse applicaties. Daarbij is het juist van belang dat de integriteit van de data gewaarborgd wordt door te zorgen dat de business rules bij connecties met de database altijd worden nageleefd. Daarnaast is het verstandig te kijken naar de wijze waarop de integriteit en beveiliging van die gegevens is georganiseerd en de totale beheerslasten. Angst van uw IT-organisatie voor een nieuwe, vreemde database is prima weg te nemen door te kiezen voor een beheerloze variant of over het databasebeheer afspraken te maken met uw pakketleverancier. ■

Bart Groothuis is adviseur bij M&I/Partners en Jurrie Zaat directeur van Orcado.