‘Niemand is schuldig, behalve de dba’

Anjo Kolk wil ‘application efficiency’

Anjo Kolk – sinds kort een auteur van Optimize – is een echte Oracle veteraan. Hij is er in 1985 begonnen en pas zestien later verliet hij het – ondertussen wel heel erg veranderde – bedrijf. In dit interview vertelt hij over de beginjaren bij Oracle, de problemen bij grote Oracle-installaties die hij moest oplossen, performance problemen in het algemeen, de verantwoordelijkheden van de dba daarin en de veranderende rol van de dba.

Nog tijdens zijn HEAO-informatica kwam Anjo terecht bij Oracle. Toch was dat geen bewuste keuze: Anjo’s voorkeur voor een stageplaats – want daar ging het aanvankelijk om – had Victoria Vesta, maar daar was geen plaats op dat moment. Er was wel plaats bij een bedrijf in Naarden, dat iets met relationele databases deden, met SQL, waar in C geprogrammeerd werd en dat werkte met user-interfaces. Anjo besloot dat dat de toekomst was en maakte een afspraak.

Kolk: ‘Ik kwam bij een kasteeltje in Naarden-Bussum, alwaar Gerry McGuire mij opwachtte. Niemand had me verteld dat het gesprek in het Engels gevoerd zou worden. Gerry liet me een schermpje zien: UFI, dat was een programma toentertijd in Oracle versie 4, dus ik zei UFI, dat is User Friendly Interface waarop hij antwoordde: heel goed, op de manier van: ‘je mag meteen blijven’.

Een lachertje

Anjo liep er stage en begon na zijn eindexamen meteen bij Oracle.

Kolk: ‘Eerst zat ik bij Jesper Larssen. Hij is verantwoordelijk geweest voor het NLS support. Mijn eerste projectje was om een portable library manual te schrijven, voor SQLmanual. Maar op de HEAO bedrijfsinformatica werd je niet onderricht in het programmeren, alleen in informatieanalyse en in het runnen van projecten. Het begon eigenlijk zo: ‘Het moet in C’. ‘C?’ Dus ik zat elke dag in de trein naar Amersfoort dat boek over C maar te lezen. Toen ben ik gaan schrijven. Dat is ook een product geworden toentertijd, en het ging op een gegeven moment eigenlijk best wel goed. Hartstikke leuk, maar ook een harde leerschool met Jesper Larssen achter je broek aan. Oracle was toen nog klein, met een omzet van 13 miljoen dollar wereldwijd en dat verdubbelde ieder jaar. De toestanden die je toen meemaakte, wat allemaal kon was ongelofelijk. Ik zat toen op de ontwikkelafdeling en heb toen geruild met iemand van de porting afdeling. Porting bestond helemaal niet. De problemen die je ook kreeg waren compilers die niet werkten. Unix bijvoorbeeld was nieuw. In die tijd lag de focus ook iets meer op functionaliteit dan op kwaliteit, dus je was heel veel bugs aan het fixen. Toen hebben we mede door Europa de kwaliteit van het product kunnen verbeteren. Het grappige met versie vier en vijf was: de performance was niet belangrijk, functionaliteit wel, je kon met TCP/IP toentertijd niet meer dan 2.5 of 5 transacties per seconde halen vanwege de architectuur van het product. Ging je naar een SAP-machine, de eerste sequentie met Oracle 5 was een lachertje, hetzelfde met de Siemens machines waar acht CPU’s in zaten: dat maakte niets uit.’

> ***Dus ik elke dag in de trein naar Amersfoort dat boek over C maar lezen***

Pas bij versie zes begon performance een rol te spelen. De bodem van Oracle werd toen compleet herschreven. Kolk: ‘In het begin waren er veel problemen om het stabiel te krijgen, 6.033, 6.034 waren de eerste versies waar een beetje stabiliteit in zat. Dat was eind jaren tachtig. Tegelijk met de steeds belangrijker wordende performance, werden de machines steeds ingewikkelder. We zijn met OS mensen in gesprek geweest van NCR en IBM en hebben veranderingen in het OS voorgesteld, bijvoorbeeld asynch IO wat niet of haast niet bestond, en hoe dat geïmplementeerd kon worden voor Oracle, nieuwe scheduling systemen.’

Mailen naar Ellison

De charme van de beginperiode was vooral dat Kolk veel verschillende dingen deed, en eigenlijk vaak aan zijn eigen lot overgelaten werd.

'Kolk: Alles kon, je werd overal voor ingezet. In Zwitserland was er bijvoorbeeld een klant die een NCR-machine had, en alles in Cobol schreef. Alleen, er was geen interface van Cobol naar Oracle. Dus werd er tegen mij gezegd: 'stap maar op het vliegtuig en ga maar een interface maken. Hoe doe ik dat? Zoek het maar uit. Dan stap je op het vliegtuig en dan lukt dat op een gegeven moment wel. Je kennis van het product Oracle werd dus steeds uitgebreider. Toen werd Oracle eigenlijk steeds belangrijker, Oracle versie 7 kwam uit, parallel server ging de markt op. 6.036 had al parallel server, dus er moesten lock managers geïmplementeerd worden op Unix systemen, dat deden we ook in Nederland.'

Kolk: 'In 1992 kwam Your Car, een van de grootste autoverhuurders, naar ons toe. Zij wilden een reserveringssyteem bouwen op Oracle, alle terminals van alle vliegvelden en zo moesten allemaal verbonden worden met Parijs. Drie weken later draait het. Dat is de eerste keer dat we ruim tweeduizend gebruikers getest hebben. Dan stuur je een mailtje naar Larry Ellison en komt er meteen een reply: geweldig jongens. Eén van de dingen die zij verkeerd deden, was dat wanneer je naar Heathrow naar de checkin voor de auto ging, en je vroeg: is er een auto voor mij? Dan drukte er iemand op een knopje en werd er een query op de database gedaan en alle auto's die op het parkeerterrein stonden, kwamen terug. Moet je voorstellen hoeveel auto's er op Londen Heathrow staan en dat moest allemaal uit Parijs komen via het netwerk. Dus dan kun je nagaan wat dat voor een impact heeft op de performance. Nu komt alleen maar de toptien van de auto's die in die klasse vallen eruit. Dat waren allemaal van die proof of concept dingen die je gaat doen: de hardware die steeds kapot gaat, het OS wat niet goed werkt, Oracle dat hier en daar uit de bocht vliegt. Dat moet je dan maar bij elkaar zien te houden. Nu ben je vooral bezig om een heel systeem in elkaar te zetten. Dan zie je op een gegeven moment wel hoe het moet.'

Dan stuur je een mailtje naar Larry Ellison en komt er meteen een reply: 'geweldig jongens'

Blank jochie

Na een aantal jaren tijdje was Anjo toe aan andere uitdagingen. Hij ging in 1996 naar Japan gegaan waar hij één jaar bleef.

Kolk: 'NTT Dokomo is een van de grootste mobiele telefoonmaatschappijen ter wereld, ze hebben nu een partnership met KPN voor i-mode. Hun hele backoffice is op Oracle geïmplementeerd. Ze hebben iets van 30-35 miljoen abonnees, die zitten allemaal in de Oracle database. Het is een van de grootste Oracle systemen in de wereld. Toen ik er kwam draaide dat systeem helemaal niet goed. Volgens NTT Dokomo lag het aan Parallel Server en Oracle. Toen we begonnen konden we zes transacties per minuut doen. Een van de laatste keren dat ik erbij betrokken ben geweest twee of drie jaar geleden hebben we het ding getest in Amerika, en toen konden we – dacht ik - 25.000 of 50.000 transacties per minuut doen. Nieuwe hardware natuurlijk, maar het systeem was zodanig ontworpen dat het er ook gebruik van kon maken. Het was gewoon scalable. Ik heb gewoon een jaar lang nodig gehad om die Japanners ervan te overtuigen dat mijn visie de juiste was. Ik moest hun vertrouwen winnen. Maar nu, als ik naar Japan toe ga en ik stap daar binnen, word ik daar ontvangen met alle égards.'

Maar waarom wilden ze dat niet geloven?

Kolk. 'Performanceproblemen zijn niet technisch, die zijn cultureel, politiek, psychologisch, noem maar op. Dat heeft niets met techniek te maken. Als ik ergens binnen loop en ik kan zien dat er iets mis is, en ik vertel de mensen wat er mis is, en de mensen werken er al twee jaar aan en hebben het nog niet gezien, dan willen ze dat niet accepteren. Als je dus met een team van driehonderd man ergens aan gewerkt hebt en dan komt er zo'n blank jochie uit Californië aan, en die zal wel eens even vertellen wat er mis, dan kan dat niet.'

Hardware

Na een jaar in Japan vertrok Kolk naar Oracle's performance groep in Amerika. Hij zag steeds meer systemen, en steeds vaker wat mensen verkeerd deden. Volgens Kolk is dat heel veel.

Kolk: 'Ik heb wel eens gezegd dat tachtig procent van de Oracle installaties in mijn beleving niet optimaal draait. Tegenwoordig wordt er veel hardware gekocht om het draaiend te krijgen. In Amerika ben ik bij een treinmaatschappij betrokken geraakt en daar hadden ze een systeem van 16 cpu's. Je kent die containers wel in Amerika, die zijn ontzettend groot en die treinen ontzettend lang. Het systeem draaide natuurlijk niet goed, en daarom werd mij gevraagd te gaan kijken. Men dacht dat er een bug in de databasewaarden zat. Ik zag al snel dat er geen bug in zat, dus ging ik tunen. Dan zie je dat die mensen dingen doen, die nergens op slaan. Bijvoorbeeld: iedere keer als er een pakketje in die container werd gezet, werd de totale prijs van de container

berekend. Zou je dat niet beter doen, wanneer de container vol is of de deuren dicht gaan, denk ik dan? Aan het einde van de rit was het zo dat het hele systeem acht keer sneller draaide op acht cpu's. Bij de vertaling van het business proces naar de IT-oplossing was van alles verkeerd gegaan, zodat ze iets gebouwd hadden, wat totaal onlogisch was. Dat is nog steeds zo. Met i-mode kun je de mails van je pop server lezen. Het grote probleem in Japan is, dat iedereen 's ochtends in de subway email gaat lezen, en dat dan de emailserver plat. Hoe los je dat op? Meer hardware? Nee, gewoon hier en daar wat vertraging inbouwen, in plaats van dat iedereen regelrecht op de server terechtkomt, gooi je queuing erin en dan werkt het. Maar het gekke is dat iedereen er zo diep in zit dat niemand meer in staat is een stap terug te doen en te kijken van: wat doen we nu eigenlijk jongens. Heel vaak als je als nuchtere Hollander, zonder dat je weet wat het systeem doet, vragen stelt, kun je al een bewustwording creëren. Dat was ook eigenlijk mijn rol bij Oracle: troubleshooten op de systemen.'

Checkpoints

Kolk: 'Wat ik gemerkt heb is dat ik vroeger op bitniveau naar het operating systeem zat te kijken, nu neem ik gewoon een stap eruit en ik kijk naar beneden en zeg: dit klopt niet, moet je eens kijken wat je aan het doen bent. De meeste mensen kijken hoe duur een SQL-statement is. Het eerste waarnaar je moet kijken is, of je het SQL-statement wel nodig hebt. Laatst vertelde ik dat op een lezing. Twee dagen later ontving ik een e-mail: 'Bedankt voor de tip, ik heb het toegepast en een batchjob die twee uur draaide, is nu klaar in tien minuten'. Er was geen enkel init.ora parameter, geen enkel snel SQL-statement wat ik op het bord geschreven heb, alleen een formule: hoe vaak doe ik iets en hoeveel kost het per keer. Maar wat heel veel dba's willen is een snelle oplossing; ik heb een int.ora parameter nodig, ik heb een quickfix nodig zonder naar het werkelijke moment te kijken.'

Hardware kan zelfs een goed werkend systeem in de weg staan. Kolk. 'Een paar weken geleden een klant van ons in Nederland: die mensen deden 15.000 checkpoints per uur, vier per seconde, echt te gek. Die mensen hebben zelfs nog grotere of meer hardware gekocht, om het hele zaakje te kunnen draaien. Dus ik adviseerde: die parameter moet je zo zetten, en dat zo, en daar moet je een index bouwen en dan zal het een stuk beter gaan. Maar dat doen ze dan toch niet, want ze hebben net gezegd dat ze een nieuwe machine moesten kopen. Ik heb een Italiaanse telefoonmaatschappij meegemaakt, die gingen naar een 32 cpu Alpha machine van Digital, want ze hadden performance problemen. Wat gebeurde er: het probleem werd erger. Het was namelijk niet de machine waar het probleem zat, maar het netwerk. De grap was, dat die man helemaal in paniek was, want die cpu's werden helemaal niet gebruikt. Er moest meer cpu-utilisation zijn! Ik heb gewoon een parameter verhoogd, die ervoor zorgt dat er meer gespind wordt, en zijn grafiekjes zagen er mooi uit waarmee hij naar het management kon gaan.'

'Bedankt voor de tip, ik heb het toegepast en een batchjob die twee uur draaide, is nu klaar in tien minuten

Trucjes

Volgens Anjo Kolk hebben dba's het in ieder geval in één opzicht gemakkelijker dan vroeger: er zijn inmiddels heel veel mogelijkheden informatie te vinden over de manier waarop je een database moet beheren. Het nut van al die informatie is volgens hem echter nogal betrekkelijk.

Kolk: 'Als je ziet wat er in die boeken staat, dat is allemaal hetzelfde. Er wordt geen enkel nieuw idee naar voren gebracht. Er worden je allemaal trucjes geleerd, in plaats van dat er een methode ligt van jongens zo pak je een performance probleem aan, of zo ontwikkel je iets, in Oracle. Daarover zijn weinig of geen boeken.

Waarom niet?

Kolk: 'Een Amerikaans spreekwoord zegt: 'learn somebody how to fish, so he can feed himself.' In Nederland verzuipen dba's, die zijn niet erin geïnteresseerd om te leren zwemmen, willen een reddingsboot. Dat is de ene kant. Aan de andere kant: al die boeken, wanneer je maar een paar bladzijdes open slaat, dan zie je: tip, tip, tip. Dat is het ergste wat iemand kan gebeuren. Want wat doet die dba? Die gaat ernaar kijken, en die ziet dan van: hé, dat is een leuke tip, laat ik het eens proberen, hé dat werkt inderdaad bij ons. Als ik dat doe, dan wordt dat getalletje beter. Maar hij vraagt zich niet af: en de eindgebruiker dan, heeft die er last van? Nee, maar waarom ga je dan de database plat leggen om het te veranderen. Ik heb wel bij vergaderingen gezeten, van Union Pacific, crisis meeting, performance deugt niet, wat doen we eraan. Dan reken ik die mensen voor: 80% van de problemen ligt hier, en u wilt 50% performance verbetering boeken, dus we moeten dat probleem aanpakken. Dan staat er een goochemerd op en die zegt: oké, daar kunnen we nu eventjes niets aan doen, kunnen we die 50% niet ergens anders vandaan halen?

Dan kijk je zo'n man aan en dan denk je: hij heeft het gewoon niet begrepen. Als 80% daar ligt, en ik heb nog maar 20% over, waar haal ik dan mijn 50% vandaan? Dan geef je die dingen aan, en soms valt het kwartje meteen, maar soms duurt het weken voordat ze denken: 'ja, we moeten het toch maar doen'.
Maar heel vaak is het zo dat mensen geen methode hebben om een probleem te analyseren en een plaatje voor het management neer te leggen, van: 'luister eens, hier zit het, we moeten dit doen'. Wat ze allemaal doen, is die trucjes toepassen. Ik noem dat checklist tuning, maar als jouw probleem duidelijk wordt bij het laatste punt op de checklist, wanneer kom je dan bij het laatste punt terecht? Hoe lang gaat dat duren? En ook

Tegenwoordig wordt er veel hardware gekocht om het draaiend te krijgen

als je het aanpakt, hoeveel procent verbetering ga jij brengen? De meeste dba's kunnen je dat helemaal niet zeggen, die zitten echt in het luchtledige te schieten. Soms hebben ze vijf procent verbetering nodig en dan veranderen ze iets en de database gaat plat en weer omhoog. Dan heeft het niet geholpen. Maar de database is plat gegaan, de applicatie heeft niet kunnen draaien. Ik ben nu bezig met een probleem bij BMW, dat is een lopende band met alle auto's die gefabriceerd worden, die komen allemaal in die database terecht. Al die veranderingen voor die auto's worden allemaal centraal bijgehouden. Dat systeem mag niet plat want dan staat het stil, dus was doe je? Hoe ga je uitzoeken waar het probleem zit? Niet aan de hand van die checklist. Dat werkt niet, je moet een methode hebben.

Performance-problemen zijn niet technisch, die zijn cultureel, politiek, psychologisch, noem maar op

Holy smoke

Eén van de laatste dingen die ik gedaan heb bij Oracle, is het ontwikkelen van een methode waarmee mensen een grafiekje kunnen maken: zoveel procent cpu, zoveel procent wachttijd. Hé mijn wachttijd is groter, waar zit die wachttijd? Als ik hier wat terugpak en ik pak daar wat terug op die en die gebieden en ik doe het zo en zo, dan kom ik heel ver. Dat is response tijd model analyseren, maar het grappige is dat ik dat bij NTT Dokomo in 1996 heb toegepast. De scripts die ik toen ontwikkeld heb worden nog steeds gebruikt. Wat is er nu eigenlijk aan de gang in het Oracle wereldje: dat wait-based /response tijd gebaseerde tuning, dat is nu the hottest thing. De meeste presentaties bij de IOG vorige week in San Diego gingen erover en die werden ook hartstikke goed bezocht dus nu beginnen ze het pas op te pakken. Maar alle boeken en cursussen zitten nog steeds in dat oude denkwereldje. We leiden nog steeds dba's op die niet kunnen voorspellen wat er met het systeem gaat gebeuren. Dat is als je er goed over nadenkt een beetje absurd, maar we accepteren het.'

De methode die Anjo ontwikkeld had om performance problemen te analyseren, had hij op een website gezet. Zijn huidige werkgever, Precise, stuitte daar op en nam contact met hem op. Kolk: 'Ik heb naar het product van dat bedrijf zitten kijken en dacht: "dat kan ik ook, dat is niets nieuws", totdat ik na een half uurtje naar het product gekeken had, en dacht: "holy smoke, dit kan ik niet, wat ze hier doen is ontzettend slim"'. In mijn huidige rol zorg ik ervoor dat er nieuwe functionaliteit in Precise ingebouwd wordt met betrekking tot Oracle. Een hoop van die ideeën die ik over performance heb, kan ik daarin kwijt. Wij hebben nu al functionaliteit waardoor je kunt zien wat er gebeurt als je ergens een index op gooit, of andere queries daar niet de dupe van worden bijvoorbeeld. De functionaliteit die is er al, maar bijvoorbeeld wat als ik nu een grote buffer cache heb, wat gebeurt er als ik dit statement zo schrijf, dat zijn dingen waarmee we nu bezig zijn omdat te gaan uitzoeken. Mijn insteek is weer wat er gebeurt op instance-niveau. Niet zozeer naar de SQL-statements kijken maar gewoon: de dba een lijst te geven van dit heeft zoveel procent van slagen als je dit verandert. Dan kan ik misschien ook nog een scriptje voor die man maken, dat dan gegenereerd wordt, daar zijn we al mee bezig.'

Alarmbellen

Een dba is vaak een weinig benijdenswaardig persoon, volgens Kolk: 'Als je een databaseserver, een applicatieserver, webserver hebt, noem maar op hebt, en het zo slecht draait in die hele stack, dan wordt er gezegd: een ding weten we, het is de database, we hoeven er niet eens naar te kijken. Waarom? Omdat de database vrijwel altijd de meeste informatie geeft. Maar een SQL-statement wordt niet door de database bedacht, maar door de applicatie. Dat zie je dat mensen steeds meer naar de database wijzen, zonder dat ze naar de rest van de stacks kijken: 'niemand is schuldig, behalve de dba.' En dan is het vaak ook nog zo dat de dba een persoon is, die hele gemakkelijk als pispaaltje kan dienen. Ik vroeg bij een project in China naar informatie: we hebben een mooie digital machine en daar hebben we de database op staan en dan hebben we vijftig applicatieservers staan (over heel China verdeeld, grote en kleine), waarop ik vroeg: wat draaien die applicatieservers dan? Allemaal op Windows 2000 en dan was er iets geschreven met VB. Bij mij gingen de alarmbellen meteen al af, want als ik VB hoor, dan zie ik al performance problemen. Niet omdat VB traag is, maar omdat wat mensen op het scherm kunnen tekenen, vaak meer

Dan staat er een goochemerd op en die zegt: oké, daar kunnen we nu eventjes niets aan doen, kunnen we die 50% niet ergens anders vandaan halen?

52: Advertentie RADventure

om de mooiigheid dan om de functionaliteit gaat. Dus zei ik tegen die projectleider: je moet oppassen dat je die programmeurs in de touw houdt. Voordat je het weet kunnen ze een pulldown list doen en ze halen alle rijen over die ze helemaal niet nodig hebben. Toen zei hij: 'yes, we had the same problem already here'. Dat zie je steeds vaker, met dat hele net- en .Net-gebeuren. Ik denk dat er steeds meer een laag komt, waarin degene die ontwikkelt niet meer weet waarvoor hij ontwikkelt. Het kan heel goed werken voor project a maar nu gaan we naar project b toe en het is te langzaam omdat het er helemaal niet voor bedoeld is.'

Ontwikkelingen

Kolk: 'Ik denk dat er twee ontwikkelingen zijn: ten eerste dat Oracle waarschijnlijk steeds meer self-tuning gaat worden, en self managing, wat je nu al ziet met locally managed table spaces, system managed undersegments en dat soort zaken. Vele van de taken die een dba uitvoert, die verdwijnen toch al. Voor een dba wordt het nu zaak om te zeggen, moet ik daar blijven hangen of moei ik gaan kijken. Ik denk dat een dba in plaats van naar de database te blijven kijken, zich moet omdraaien en de stack op moet kijken, van wat wordt er naar me toe gegooid en waar zit het allemaal en ik heb daarover al met een aantal heren over zitten praten en wij hebben de term application-efficiency bedacht. Geef twee mensen een business probleem en laat ze een oplossing bedenken; ze doen allebei hetzelfde, maar welke van de twee is het meest efficiënt? Op dit moment kunnen we wel zeggen hoe een database performt, maar we kunnen niet zeggen, hoe een applicatie performt, in termen van: deze is zo efficiënt en deze zo. Dan kun je van tevoren aangeven: we kunnen je hardware requirement met die factor terugbrengen. Ik denk dat de dba steeds meer naar de applicatie-efficiency moet gaan kijken. Dat daar de tools voor beschikbaar moeten worden die dat in de gaten gaan houden, die het business gebeuren in de gaten gaan houden, en niet gaan kijken: is mijn buffercache hitratio wel goed. Wie interesseert dat nou? Wanneer die dba toch de schuld krijgt, dan kan hij zich er maar beter echt mee gaan bemoeien.'

We zitten nog steeds dba's op te leiden die niet kunnen voorspellen wat er met het systeem gaat gebeuren

Soms duurt het weken voordat ze denken: 'ja, we moeten het toch maar doen'

Dré de Man