

Oracle Designer 6i

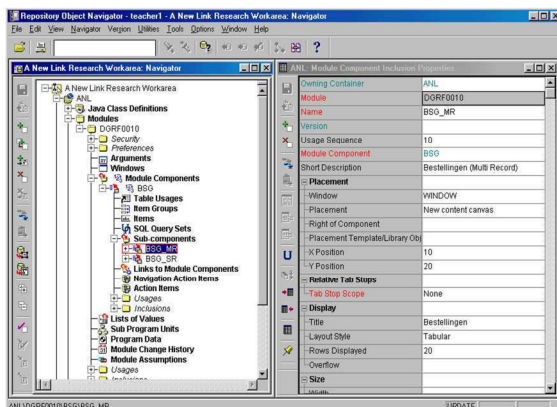
De andere voordelen

In de vorige *Optimize* hebben we een begin gemaakt met de beschrijving van Oracle Designer 6i. Daarbij hebben we ons vooral gericht op de mogelijkheden van Designer met betrekking tot versiebeheer en autorisatie. In dit artikel zullen we aandacht besteden aan het genereren van Forms, het gebruik van Headstart in combinatie met Designer 6i en het migreren van applicaties van vorige releases van Designer naar Designer 6i.

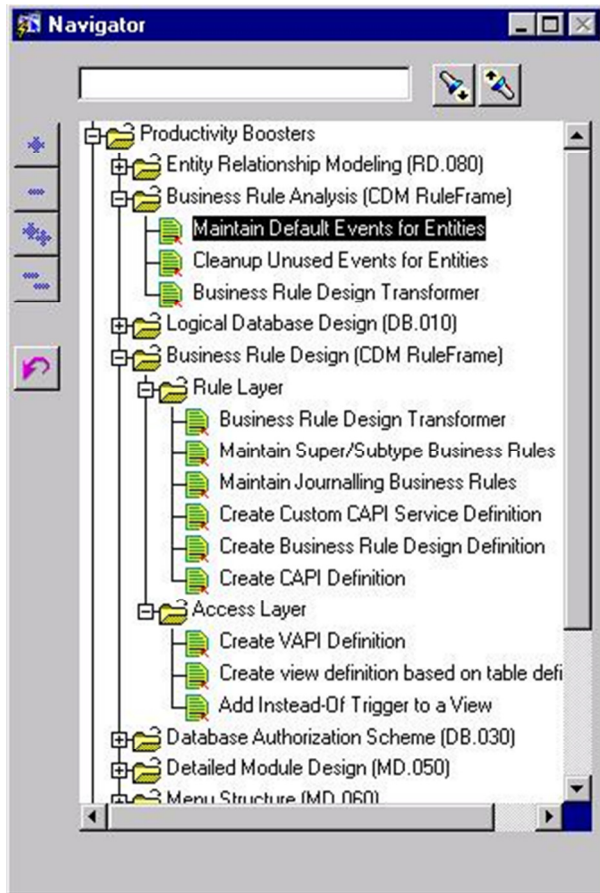
Veruit de belangrijkste reden voor organisaties om Oracle Designer 6i te gaan gebruiken is het scala aan mogelijkheden dat het product biedt met betrekking tot configuratiemanagement. Als dit echter de enige reden is om voor Designer 6i te kiezen, doen we het toch danig tekort. Immers, Designer is op veel meer punten verbeterd. Zo is bijvoorbeeld de bijbehorende Forms Generator veel rijker geworden qua functionaliteit. Om dit te illustreren zullen we verschillende mogelijkheden met betrekking tot het genereren van Oracle Forms kort beschrijven. Aan de orde komen achtereenvolgens: multi-region blocks, navigatortrees, (reusable) LOV-objects en de Relative Tab Stops Editor.

Multi-region blocks

Zoals we in het vorige artikel over Oracle Designer 6i al hebben kunnen lezen, wordt de functionaliteit van de Oracle Repository en Oracle Designer in belangrijke mate bepaald door ontwikkelingen rond Oracle Applications. Zo vindt ook het begrip 'Multi-region blocks' z'n oorsprong in Applications. Dergelijke blokken worden daar aangeduid als 'Combination Blocks'. Maar wat is dat nu, een multi-region block? Een multi-region block is een blok waarbij de items niet slechts beperkt zijn tot één vierhoekig gebied op één canvas en één window. Dus hoewel een multi-region blok is gebaseerd op één enkele tabel, kan hij versnipperd worden weergegeven, op een aantal vierhoekige gebieden. Daarbij mogen die gebieden ('regions') worden geplaatst op hetzelfde canvas, op verschillende canvases in hetzelfde window en op verschillende canvases in verschillende windows. De mogelijkheden zijn echter nog groter. Het aantal weergegeven records in elk van die regions kan namelijk verschillend zijn. Zo kan in de ene region worden gekozen voor een single record layout, terwijl in een andere region een multi-record weergave wordt toegepast. Daarbij zijn de verschillende regions uiteraard volledig gesynchroniseerd. Om een dergelijke layout te kunnen specificeren in Designer, is er een nieuw elementtype bijgekomen: de subcomponent. Voortaan kan in Designer een Module Component worden verfijnd in Module Sub-Components. Elk van deze Sub-Components is de basis voor een van de regions van het Multi-region block. In afbeelding 1 is te zien dat in Designer de properties van Module Sub-Component in feite vergelijkbaar zijn met die van Module Components.



Afbeelding 1 Properties van Sub-Components

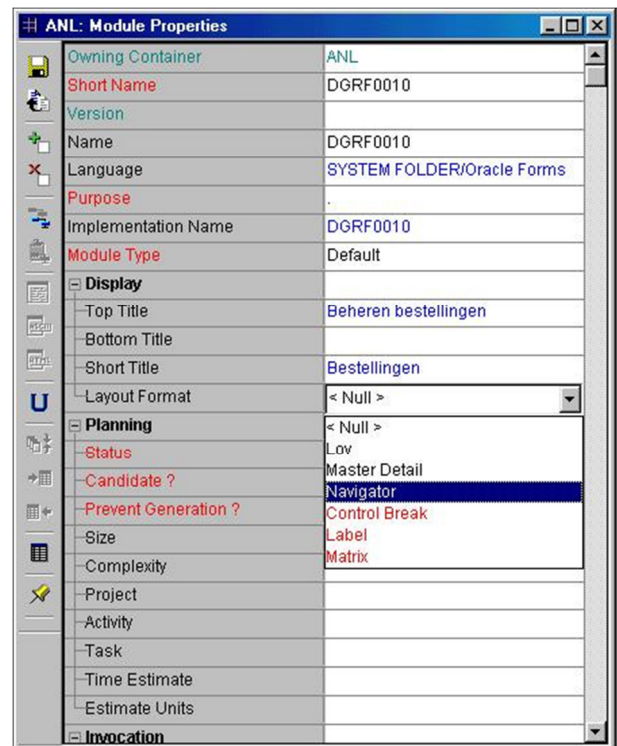


Afbeelding 2 Layoutvoorbeeld van een genereerbare navigator tree

Navigator Trees

In moderne layouts lijkt het een waar modeartikel te zijn geworden: de navigator tree. Waarschijnlijk liggen de roots van dit fenomeen in het basisgereedschap van Microsoft Windows, de Explorer, of nog verder terug, de File Manager. Door hiërarchisch gerelateerde gegevens in een boomstructuur weer te geven kunnen nodes worden 'opengeklapt' en kan net zo lang worden 'afgedaald' in de structuur tot het gewenste gegeven is gevonden. Deze layoutvorm is overigens niet geschikt voor elke gegevensstructuur. Allereerst moeten de gegevens zoals gezegd hiërarchisch gerelateerd zijn. Dat houdt in dat de weer te geven records moeten komen uit tabellen die een master-detail relatie met elkaar onderhouden. Daarnaast is het een vereiste dat het aantal details dat hoort bij een master niet al te groot is. Immers, zodra een node opengeklapt wordt waarbij er een duizendtal of meer records zichtbaar worden, dan leidt het scrollen door die vele records ertoe dat de hiërarchische structuur uit het zicht verdwijnt, en het overzicht op de structuur volledig verdwijnt. Het doel van de navigator tree is uiteraard navigatie. Dat wil zeggen dat zodra het gezochte gegeven is

gevonden, dat dan door het selecteren daarvan er een tweede window moet openen waar het geselecteerde gegeven in detail (single record block) wordt weergegeven. In feite is de dialoog van de Headstart Utilities ook een voorbeeld van een navigator tree en een bijbehorend detailwindow (zie afbeelding 2). Een dergelijke tree is al langer te bouwen met Oracle Forms; al vanaf Oracle Forms 6.0 is er native ondersteuning van tree items. Forms met navigator trees vormen een vast onderdeel van de demoforms die meegeleverd worden met Oracle Forms 6.0. Vanaf Oracle Designer 6i is het echter ook mogelijk dergelijke layouts te genereren. Met betrekking tot navigator trees kunnen er drie varianten worden vastgelegd in Designer, variërend van een eenvoudig vast te leggen variant met een grote, niet wijzbare, standaardfunctionaliteit, tot een bewerkelijker, maar controleerbaardere variant. De varianten zijn achtereenvolgens:

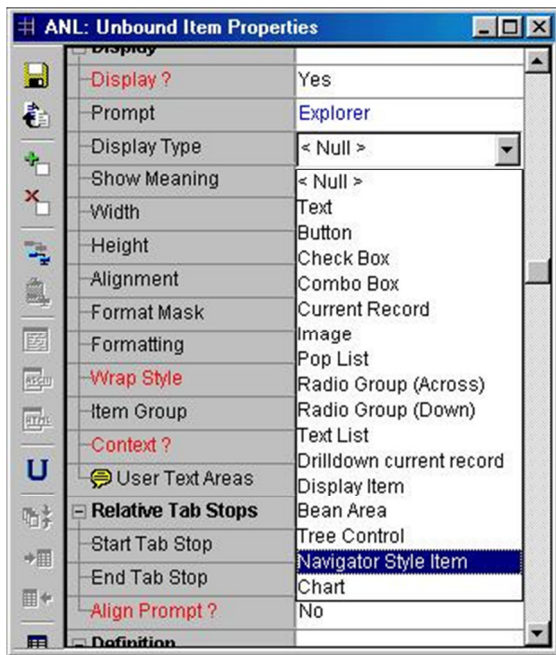


Afbeelding 3a Het definiëren van een navigator style form

Navigator Style Forms

Het definiëren van een Navigator Style Form is erg eenvoudig. Voor het tree-gedeelte behoeft eigenlijk niets extra te worden gedaan. De ontwikkelaar specificeert de modulecomponents -waarmee in detail de gegevens onderhouden kunnen worden waarnaar genavigeerd wordt- en zet het Layout Format property van de module op 'Navigator' (zie afbeelding 3a). Hierdoor wordt door de Forms Generator een extra window gegenereerd (het

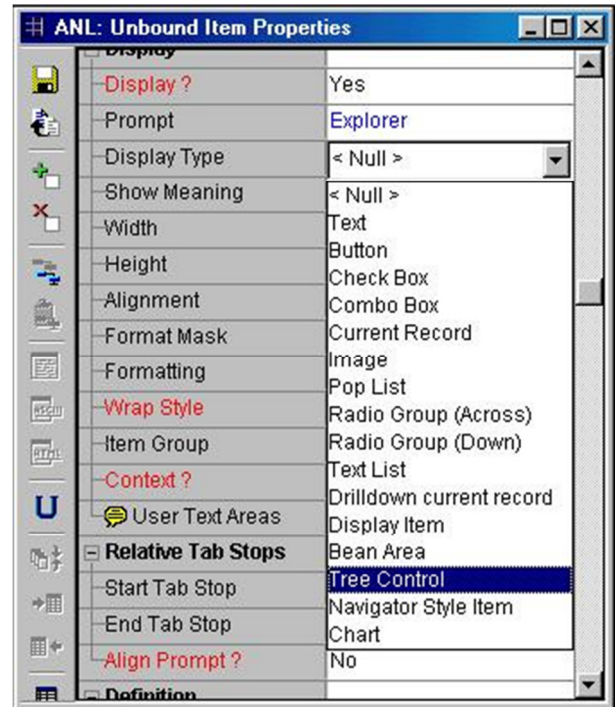
eerste window) met daarin een navigator tree, waarvoor dus geen directe definitie terug te vinden is in Designer. Hierin worden de mastertabellen uit de Module Components gepresenteerd als de main nodes van de tree. Forms Generator genereert zelf de code die nodig is om de tree-items te populieren met data. De enige manier om het gegenereerde navigatorwindow en bijbehorende canvas te beïnvloeden is het daarvoor gebruikte standaardobject in de objectlibrary aan te passen.



Afbeelding 3b Het definiëren van een navigator style item

Navigator Style Items

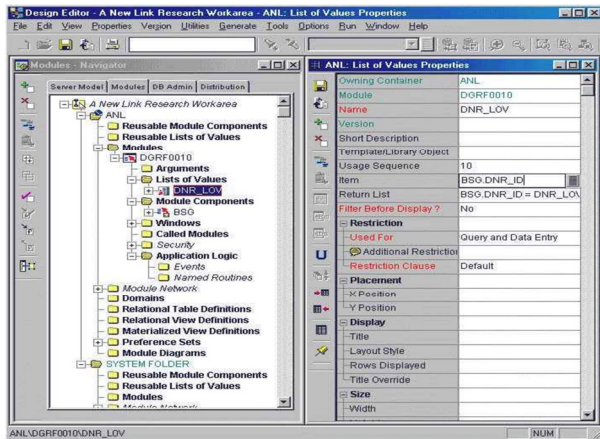
Bij het definiëren van Navigator Style Items is de ontwikkelaar al veel nadrukkelijker bezig met het specificeren van de tree. Voor de plaatsen waar de trees moeten komen dienen de gebieden op de canvases gedefinieerd te worden (door het gebruik van Module Component en Module Sub-Component; zie Multi-region blocks) en dient het te gebruiken window opgegeven te worden. Vervolgens moeten de items van het type 'Unbound item' worden gekozen, met als Display Type 'Navigator Style Item' (zie afbeelding 3b). Forms Generator genereert zelf de code die nodig is om de tree-items te populieren met data. In de repository kunnen de properties worden onderhouden van de items en de gebruikte canvases en windows.



Afbeelding 3c Het definiëren van een hierarchical tree item

Hierarchical Tree Items

Items van het type 'Hierarchical Tree Items' leiden tot het soort tree-item met de hoogste mate van 'zelfbeschikking'. Alles wat opgaat voor Navigator Style Items gaat ook op voor deze items, met één verschil: de ontwikkelaar dient zelf de applicatielogica te schrijven om de tree-items te populieren met data. Hiervoor zijn verschillende events in Forms opgenomen waarop de logica kan worden getriggert: bijvoorbeeld de WHEN-NEW-FORM-INSTANCE om de tree te populieren, en WHEN-TREE-NODE-ACTIVATED, WHEN-TREE-NODE-EXPANDED en WHEN-TREE-NODE-SELECTED om eventueel extra manipulaties uit te kunnen voeren. Het manipuleren zelf kan worden gedaan met de Forms built-in package 'Ftree', met daarin functies zoals `add_tree_data()`, `delete_tree_node()`, `find_tree_node()`, `get_tree_node_parent()` enzovoort. Overigens moeten evenals bij Navigator Style Items de items van het type 'Unbound item' worden gekozen, met als Display Type 'Tree Control' (zie afbeelding 3c).



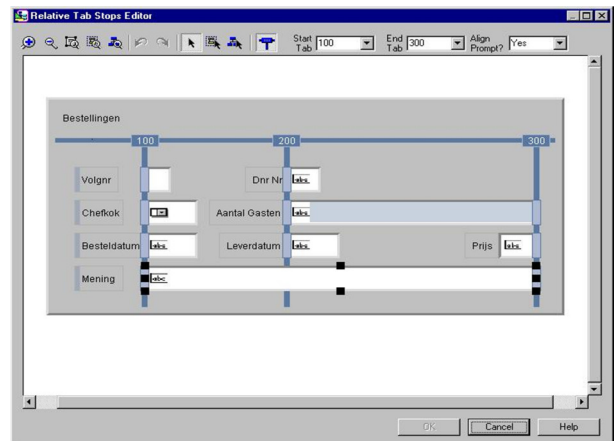
Afbeelding 4 Het definiëren van (Reusable) LOV's

(Reusable) LOV's

Al vanaf de eerste versies van CASE*Dictionary (de verre voorouder van het huidige Oracle Designer 6i) leggen we de List Of Values (LOV) op dezelfde wijze vast. Of beter gezegd: leggen we die op dezelfde wijze niet vast. Tot vóór Oracle Designer 6i ontstond de LOV bij het genereren van Oracle Forms immers als het ware 'vanzelf'. Zonder dat we in Designer een LOV object vastlegden, wist de generator op de één of andere manier te herleiden dat er een LOV op een bepaalde foreign key kolom gewenst was. Het algoritme dat hiervoor door de generator werd gebruikt was overigens erg eenvoudig. Zodra in een te genereren base table usage een item voorkwam die was gebaseerd op een foreign key kolom en deze kolom bovendien werd gebruikt om te linken met een eveneens te genereren lookup table usage, dan ontstond de gewenste list of values. Het gevolg hiervan was dat de definities van de module in Designer en in Oracle Forms in feite van elkaar verschilden. Er is echter bij Oracle een trend waar te nemen in de Designer/Developer combinatie, om deze producten zo dicht mogelijk bij elkaar te brengen. Zo was er vroeger geen object in Designer die het equivalent vormde van het concept 'Block' in Forms. Een ontwikkelaar die eerst met Oracle Forms werkte en later met Designer leerde werken had daardoor zeker even tijd nodig om het concept van 'Base en Lookup Table usages' in Designer te vertalen naar 'Blocks' in Forms. We hebben kunnen zien dat in Oracle Designer 2.1 hiervoor het concept 'Module Component' (MC) is ingevoerd. Dit objecttype is uiteraard primair geïntroduceerd om op dit nieuwe niveau functionaliteit te kunnen specificeren (zoals de mogelijkheid tot het reusable maken van een MC, het definiëren van een Datasource en -target van een MC en het kunnen genereren van een Module Component API). Het komt het begrip echter zeker ten goede dat de Module Component één op één gemapt kan worden op een Block in Oracle Forms.

Eenzelfde ontwikkeling zien we nu in Oracle Designer 6i met betrekking tot het definiëren van LOV's. Er is nu een heus objecttype 'LOV' waarmee in een module Lists of Values kunnen worden gedefinieerd (zie afbeelding 4). Op het eerste

ogenblik lijkt dit nauwelijks iets toe voegen aan de vroegere werkwijze rond LOV's. Schijn bedriegt! Doordat we een onbeperkt aantal LOV's kunnen definiëren in een module en runtime (in applicatielogica) de gegenereerde LOV's aan items kunnen koppelen gaat er een wereld aan mogelijkheden voor ontwikkelaars open. Zo kunnen we afhankelijk van een bepaalde situatie (bijvoorbeeld 'enter query-mode' of 'input-mode') de meest geschikte LOV koppelen aan een item. Evenals bij Module Components is ook voor LOV's de mogelijkheid geschapen om deze in Designer te promoveren tot een reusable variant. In dat geval overstijgt de LOV de grenzen van een Module en is deze ook in andere Modules toe te passen.



Afbeelding 5 De Relative Tab Stops Editor

Relative Tab Stops Editor

Het door ontwikkelaars meest gewaardeerde onderdeel van Oracle Designer is zonder twijfel de Relative Tab Stops Editor. Hiermee kan voor het eerst reeds in Designer elk item worden gepositioneerd op de plaats waar de ontwikkelaar dat wenst. Vergeleken hierbij kunnen we het opmaken van de layout in voorgaande versies afdoen als creatief knutselen. De grote verbetering hierbij is het concept van de Relative Tab Stops in de layouts. Dit zijn verticale tabulatorinstellingen die (in tegenstelling tot de bestaande Absolute Tab Stops) niet gerelateerd zijn aan een bepaalde x-coördinaat op de layout, maar waarvan de positie wordt bepaald door het eerste item dat met de Tab Stop uitgelijnd is. Items hebben nu twee extra properties, namelijk Start Tab en End Tab. Hiermee kunnen items repectievelijk aan het begin of aan het eind worden uitgelijnd met een bepaalde Relative Tab Stop. Het is echter onbegonnen werk om deze properties te moeten wijzigen zonder direct het effect daarvan te zien op de layout. Om daaraan tegemoet te komen is er een volledig nieuwe editor in de Design Editor ingebouwd: de Relative Tab Stops Editor. De functionaliteit van de editor -en daarmee het gehele principe van de Relative Tab Stops- kan het best worden geïllustreerd aan de hand van een voorbeeld, zoals weergegeven in afbeelding 5. Hierin is te zien dat er in het blok

'Bestellingen' een drietal tabstops is gezet, resp. met de identificatie 100, 200 en 300. Het feit dat de ruimte tussen tab 100 en 200 afwijkt van die tussen 200 en 300 geeft al aan dat het relatieve tabs zijn. De getallen hebben slechts als doel de volgorde aan te geven, en de mogelijkheid te bieden tussenliggende tabs te definiëren (maximaal 99 stuks in beide gevallen). Voorts is te zien dat items kunnen worden uitgelijnd aan het begin of het einde van het item. Zo is het item 'chefkok' geselecteerd, en is bovenin de editor als Start Tab de waarde '100' gekozen. Hiermee is het item naar de tweede regel van de layout gedirigeerd, terwijl er theoretisch gezien genoeg ruimte is op de eerste layoutregel. Alle prompts van de items van Tab Stop 100 zijn uitgelijnd door de items te selecteren en bovenin de editor Align Prompt? op de waarde 'Yes' te zetten. Soms wordt een rustiger layout verkregen door items van ongelijke lengte uit te lijnen en even lang te maken. Door in de editor bij een (korter) item zowel de Start Tab als de End Tab te zetten, treedt automatisch item expansion op, waarbij het geëxpandeerde deel in een afwijkende kleur wordt weergegeven. Dit is in het voorbeeld het geval bij item 'Aantal Gasten'.

Door het zetten van de Tab Stop Scope property van een item group of module component kan het werkingsgebied van het uitlijnen worden vastgelegd, van items binnen een item group met items buiten de item group, en van items binnen een module component met items in een andere module component.

Na enige oefening kunnen met de editor complexe layouts worden gespecificeerd in een vergelijkbare tijd als voor het specificeren met de layouteditor van Oracle Developer. Het enige minpunt van de editor is z'n wijze van benaderen. In de Design Editor kennen we bij het modelleren van modules al geruime tijd de Data View en de Display View. De Displayview was een abstracte versie van de layout zoals die op basis van de specificaties zou worden gegenereerd. Voor de hand liggend zou zijn de Relative Tab Stops Editor met de Display View te integreren. Hier is echter niet voor gekozen. Door een bepaalde component in de Display View te selecteren kan met de rechter muisknop (onder andere) de Tab Stops Editor worden opgestart. De Display View van de Design Editor verdwijnt dan geheel uit beeld en er wordt vervangen door de Tab Stops Editor. Omdat deze editor zich uitsluitend focussed op het uitlijnen van de items, verdwijnt op dat moment het overzicht op de gehele layout, zoals de Display View die wel biedt. Misschien is er bewust voor deze oplossing gekozen, maar het is wel even wennen.

Headstart

In het moederland (met waarschijnlijk ook de hoogste adoptiegraad) van het Headstart Template Package kunnen we uiteraard niet volstaan met het beschrijven van louter Oracle Designer. Welnu: Oracle Designer 6i is toe te passen in combinatie met twee versies van Headstart. Allereerst is daar Headstart 2.1 patch 13. Deze configuratie is vergelijkbaar met de Simple Headstart Migration ten tijde van de overstag van Oracle Designer 1.3.2 naar 2.1 of 6.0. Ofwel, het Template Package

onderdeel is daarmee aangepast om met de nieuwe Designer en Developer te kunnen werken, maar de (nog niet aangepaste) Headstart Utilities zijn niet conform de nieuwe structuur van de Repository API en kunnen daarom niet worden gebruikt.

De andere versie van Headstart die kan worden gebruikt is speciaal ontwikkeld voor gebruik in combinatie met Oracle Designer 6i. Dit product is niet langer een patch op een vorige Headstart versie, maar is (eindelijk) een zelfstandig product. Het gaat verschijnen onder de naam Headstart 6i. Oorspronkelijk zou deze versie al in februari beschikbaar zijn -de freeze van de code was eind januari gepland-, maar de bètatestfase is tot nader order verlengd. Wellicht is de versie ten tijde van publicatie van dit artikel wel beschikbaar. In een van de volgende uitgaven van Optimize zullen we een artikel wijden aan de functionaliteit van Headstart 6i. Om reeds een tipje van de sluier op te lichten: het wordt Oracle ernst met 'webforms'. Met Headstart 6i ontwikkelde applicaties zijn niet meer afgestemd (Oracle noemt het 'niet gecertificeerd', maar er is sowieso al nooit garantie geweest op Headstart) op gebruik in een client/server configuratie, maar zijn expliciet bedoeld om three-tier te werken. Daarnaast is Headstart zelf uitgebreid met een CGI-launch environment, waardoor ook tijdens de ontwikkeling three-tier kan worden gewerkt.

Migratie

Over migratie naar Oracle Designer 6i valt zoveel te vertellen, dat we volstaan met het aanstippen van enkele typerende zaken, en het verwijzen naar verdere literatuur. Allereerst kunnen we opmerken dat het woord migratie weloverwogen is gekozen; upgrade zou de lading te weinig dekken. Was het in vorige versies van Oracle Designer vaak nog mogelijk de repositorystructuur te upgraden of op z'n minst een archive te maken van de applicatiesystemen en deze te restoren in de nieuwe versie, met Oracle Designer 6i is dat verleden tijd. Het migratieproces dient te worden uitgevoerd van de ene up-and-running omgeving naar de andere. Anders gezegd: naast de oude Designer omgeving moet de nieuwe omgeving worden ingericht (beide repositories moeten live zijn op moment van migratie). Bij het starten van het migratieproces op de nieuwe repository zal deze via een databaselink de oude repository opschonen, controleren en vervolgens converteren en laden in de nieuwe repository. Hierbij is een opmerking op z'n plaats. Het opschonen gebeurde in de eerste versies van Oracle Designer 6i iets te rigoureuus. Met name applicaties die een migratiehistorie kenden vanaf Designer 1.3.2 moesten het ontgelden. Zo verdwenen binnen modules soms koppelingen met Module Components. Hoewel deze problemen in de recentste release (Release 2) in ieder geval lijken te zijn opgelost, blijft het migreren een arbeidsintensieve klus. Omdat de migratie-utility strenger is dan de migratie-utilities in de vorige versies, kunnen er applicatiesystemen worden afgewezen met fouten die een eerdere migratie (bijvoorbeeld van Designer 1.3.2 naar Designer 2.1) probleemloos doorstaan hebben. Het is in ieder geval aan te bevelen de nodige veiligheidsmaatregelen te treffen voor behoud van

de repository(-data) in de oude omgeving. Te denken valt aan het archiveren van applicatiesystemen, het exporteren van de gehele repositoryinstance, en van de gehele database. Zijn resources ruim voorhanden, dan is het handig om een compleet gekopieerde omgeving op te tuigen als bron voor de migratie.

Naast het inhoudelijk migreren van applicatiesystemen naar de nieuwe repository, zit een groot deel van het werk in het onderbrengen van de gegevens in 'the versioned world'. Het iDevCOE van Oracle heeft over deze materie een dikke migration guide geschreven (Oracle Designer 6i Migration Guide) waarin alle ins en outs staan over het migreren van structured en unstructured elements en überhaupt het werken in 'the versioned world'. Deze guide is te downloaden van technet, http://technet.oracle.com/products/designer/pdf/otn_des6i_migui.pdf.

Oud en nieuw

Alle mooie verhalen over voordelen van moderne softwaretools ten spijt, feit is dat we in de praktijk veelal niet te maken hebben met de ideale situatie. Er is in het verleden bijvoorbeeld een fors aantal applicaties uitgeleverd, waarvan een deel om wat voor reden dan ook nog niet gemigreerd kan worden. Daarbij komt dan bijvoorbeeld dat het beheren van zowel de reeds gemigreerde als de nog te migreren applicaties door dezelfde personen c.q. vanaf dezelfde ontwikkelwerkplekken dient te geschieden. Hierbij is het 'slecht te verkopen' dat een ontwikkelaar voor elk van de toolversies plaats moet nemen achter een andere PC. Deze situatie vraagt dus om een breed inzetbare ontwikkelwerkplek, waarmee zowel software van de oudere als de nieuwere generatie kan worden ontwikkeld. Uitgaande van een PC van het type 'Jerommeke' (recente CPU, 256+ Mb intern geheugen, 10+ Gb harddisk) is het in principe mogelijk drie generaties Designer naast elkaar te installeren en te draaien: Designer/2000 1.3.2, Oracle Designer 6.0 en Oracle Designer 6i. Het probleem hierbij is echter dat de bijbehorende Developer versies elkaar bijten. Wordt bijvoorbeeld Developer 6.0 geïnstalleerd naast Developer 1.3, dan ontstaan er problemen bij het ontwikkelen van Oracle Reports in de Developer 1.3 omgeving. Daarnaast is het überhaupt niet mogelijk om Oracle Developer 6.0 en Oracle Developer 6i naast elkaar te installeren. De reden van deze problemen is gelegen in het feit dat deze producten allemaal in dezelfde Oracle Home directory dienen te worden geïnstalleerd. Hierdoor worden sommige bestanden van de oudere generatie software per ongeluk overschreven door gelijknamige bestanden uit de jongere releases (hoezo versiebeheer?). Is dit alles onoverkomelijk? Valt mee. Er zijn twee mogelijkheden om dit probleem te tackelen. Allereerst kan er met grof geschut worden gewerkt: het bouwen van een zogenaamde registryswitcher. Op deze wijze kunnen de verschillende releases elk

in een andere homedirectory worden geïnstalleerd. Nadat een product geïnstalleerd is, dient de registryinhoud geëxporteerd te worden, en vervolgens te worden geschoond. Daarna kan een ander product in een andere homedirectory worden geïnstalleerd en de daarbij behorende registryinhoud worden veiliggesteld. Door nu op het niveau van Windows, bijvoorbeeld met een in KixStart geschreven script, de registry met de juiste registryinhoud te laden, kan de omgeving worden ingesteld voor de juiste toolversie. Nadeel van deze oplossing is dat er op enig moment maar één toolversie actief kan zijn (die waar de registry op dat moment naar verwijst).

De elegantste oplossing is echter die waarbij alle versies zonder 'omschakelen' gebruikt kunnen worden. Hiervoor is op het moment van schrijven van dit artikel maar één configuratie geschikt: allereerst dient in de oudste omgeving de bijbehorende Developer versie ge-upgrade te worden naar Developer 1.6.1. Deze versie werkt probleemloos samen met Designer/2000 1.3.2. Daarnaast moet Oracle Designer 6.0 worden gepatched met patch 6. Op die manier werkt deze versie probleemloos samen met het jongere product Oracle Developer 6i, onder de voorwaarde dat laatstgenoemde wordt gepatched met patch 1. Het behoeft geen toelichting dat ook Oracle Designer 6i uitstekend samenwerkt met Oracle Developer 6i. Gevolg van deze configuratie is uiteraard wel dat in het veld nog maar twee Developerreleases gebruikt kunnen worden: Developer 1.6.1 en 6i patch 1. De upgrade naar deze versies vanaf respectievelijk 1.3 en 6.0 is echter minimaal.

Resumerend

In een serie van twee artikelen hebben we gepoogd Oracle Designer 6i wat meer voor het voetlicht te brengen. We hebben kunnen zien dat Designer 6i meer is dan 'Designer 6.0 maar dan met versiebeheer'. Zo hebben we bij het genereren van Oracle Forms nu mogelijkheden voor generatie van de meest moderne lay-outs. Keerzijde van het werken met Oracle Designer 6i is dat het verder ontwikkelen van bestaande applicaties een migratie vereist van de repositorygegevens. Hoewel dit niet altijd een even soepel verlopend traject zal zijn, is het de moeite meer dan waard. Designer 6i is werkelijk een bijzonder fraai product. En, als extra steuntje in de rug voor de twijfelaars, support op de oudere generaties Designersoftware is zeker niet eeuwig. Dus waarom niet eerder overgestapt? Succes met de keuze!

Kijk voor meer publicaties op <http://www.anewlink.nl/ict/nl/publicaties/>.

(c) Copyright 2001 A New Link bv