

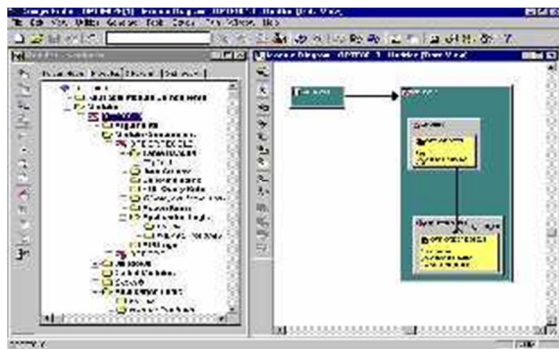
# Oracle Designer 2.1

## *Genereren vanuit de Design Editor*

*In de vorige Optimize is een begin gemaakt met een beschrijving van de nieuwe Oracle Designer, versie 2.1. In dit tweede deel wordt deze beschrijving afgerond. Met name het zogenaamde lowercase deel van Designer zal de revue passeren. Het is immers juist dit deel van Designer dat de grootste wijzigingen heeft ondergaan. De wijzigingen zijn dan ook niet zozeer een kwestie van een gewijzigde interface voor de ontwikkelaar, maar vereisen een andere manier van werken dan voorheen.*

In dit artikel zullen verschillende facetten van Designer 2.1 aan de orde komen. Qua beschrijving van het product zelf, zullen we ons echter beperken tot de meest gebruikte onderdelen in Designer 2.1 projecten. Beseft dient te worden dat Oracle Designer hierdoor eigenlijk te weinig eer aangedaan wordt. Zonder veel moeite kan een artikel met een gelijke omvang als het voorliggende, volledig worden gewijd aan bijvoorbeeld de wijzigingen met betrekking tot het genereren van PL/SQL (web-)modules voor de Oracle Application Server. Allereerst wordt de functionaliteit van de volgende onderdelen van Designer besproken: de Design Editor, de Forms Generator, de Library Generator en de Server Generator.

Vervolgens wordt beschreven wat de wijzigingen in de betreffende onderdelen van Oracle Designer voor invloed hebben op het ontwikkelproces. Het artikel wordt afgesloten met een blik in de (nabije?) toekomst met betrekking tot de functionaliteit van Oracle Designer.



Afbeelding 1 De Design Editor

### De Design Editor

Het is nog niet zo heel lang geleden dat de meeste systeemontwikkelingsprojecten volgens de traditionele watervalmethode (Classic Approach) werden uitgevoerd. Vaak deed deze aanpak echter denken aan het kanon en de mug. De sterk gefaseerde aanpak voldoet nog steeds uitstekend bij grote projecten, waarbij vele partijen betrokken zijn, sprake is van een grote doorlooptijd, en niet elke partij even ervaren is. Bij de kleinere projecten is het echter veel wenselijker een aantal fasen te combineren. Zo leidt het combineren van technisch ontwerp en bouw tot één fase (System Design and Generation) vaak tot het voorkomen van teleurstelling bij de eindgebruikersorganisatie. Het 'voortschrijdend inzicht' van onervaren eindgebruikers neemt vaak een enorme vlucht zodra het ontwikkelde informatiesysteem zichtbaar wordt. Het is daarom heel efficiënt om, zodra de klant écht weet hij wil, in enkele iteraties het technisch ontwerp en de gegenereerde software conform de verwachtingen aan te passen. Wellicht ten overvloede: ook bij de watervalmethode kunnen deze iteraties worden uitgevoerd. Alleen is er dan vaak een stortvloed aan wijzigingsvoorstellen nodig om het technisch ontwerp aan te passen. Om de ontwikkelaar beter te ondersteunen in de uitvoering van de gecombineerde systeemontwerpen en generatiefase, heeft Oracle alle onderdelen van Oracle Designer die gebruikt worden in deze fase, samengevoegd in één geïntegreerde ontwikkelomgeving: de Design Editor (zie afbeelding 1). De Design Editor herbergt een vijftal diagrammers en een zestal generatoren uit de vorige release van Oracle Designer.

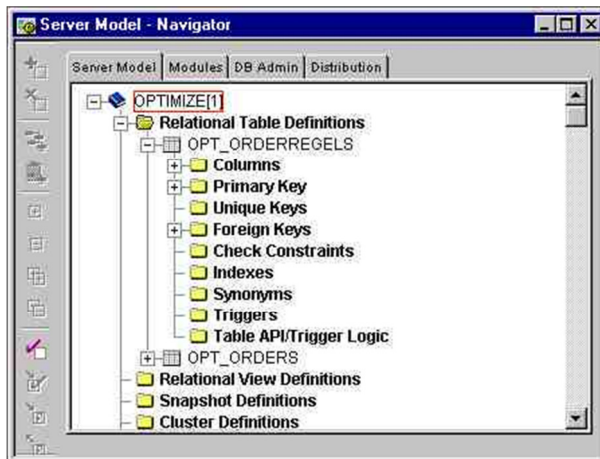
### De look-and-feel

Als bij het starten van de Design Editor het welcome screen verschijnt kunnen verschillende voorkeuren worden ingesteld. Eén van deze voorkeuren is de checkbox 'use a guide'. Door te kiezen voor deze guide wordt de gebruiker aan de hand meegenomen door de stappen van het ontwikkelproces. De onervaren gebruiker van Oracle Designer zal dit feature zeker weten te waarderen.

De Design Editor bestaat in feite uit twee onderdelen: een window met een aantal navigatorrees en een window waarin verschillende soorten diagrammen getekend kunnen worden. De keuze voor deze opzet is niet verwonderlijk. Het aantal diagrammers groeide in release 1.3.2 al aardig de pan uit. Dat betekende dat je voor een diagram van een ander soort object eerst de bijbehorende diagrammer op diende te starten. In de huidige opzet selecteer je in de navigatorree in het linker window het object van je keuze, en sleep je dat naar rechts buiten het navigatorwindow. Hierdoor wordt er een extra window geopend en wordt de juiste diagrammer opgestart om het geselecteerde object in dat window te kunnen bewerken. Het contextgevoelige menu van de Design Editor is tamelijk omvangrijk. Via dit menu, of -eindelijk conform de Windows styleguide- het menu verborgen onder de rechtermuisknop, kunnen alle generatoren en diagrammers worden opgestart.

### Het navigatorwindow

De beginnende gebruiker van release 2.1 zal nog regelmatig de fout ingaan: een object uit het navigatorwindow naar het diagrammerwindow slepen, terwijl op dat moment in de RON wordt gewerkt. Het navigatorwindow van de Design Editor heeft ook wel erg veel weg van het navigatorwindow van de RON. Het dubbelklikken van een willekeurig object leidt in beide trees tot het openen van een window met -afhankelijk van de voorkeuzeinstelling- een property palette (de vroegere property sheet) of een dialog (een soort wizard). Het meest in het oog springende verschil tussen beide tools is echter het feit dat in de Design Editor de navigatorree is gesplitst in vier onderdelen. Daarbij is elk onderdeel terug te vinden onder een eigen tab. Het betreft de volgende onderdelen:

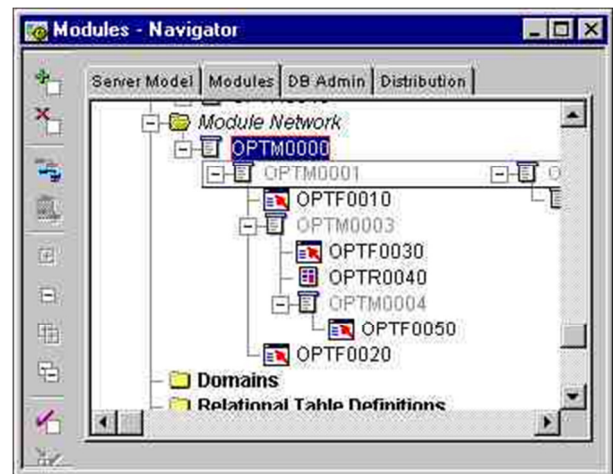


Afbeelding 2 De Server Model tab van de Design Editor

### Server Model

In de beta-releases van Oracle Designer ging dit onderdeel nog door het leven onder de titel 'DB

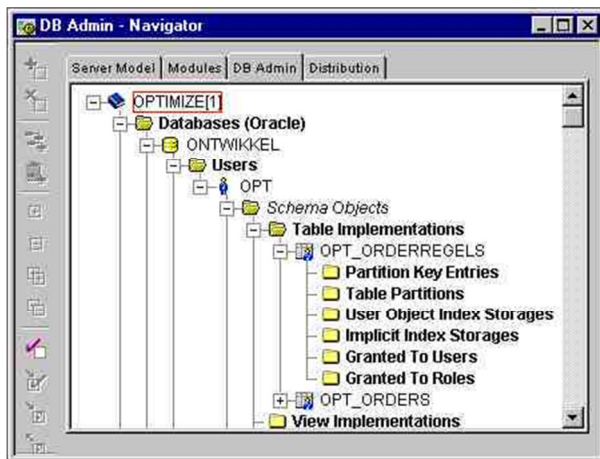
Objects'. De bitmap van de Design Editor die je ziet bij het starten van Designer getuigt hier nog van. Met het oog op de steeds belangrijker wordende Oracle Application Server is de naamswijziging niet echt een verbetering. De objecten die je onder deze tab terug vindt zijn uitsluitend objecten die fysiek in een database gecreëerd kunnen worden. Dat betekent weer even wennen voor de doorgewinterde 1.3.2. gebruiker: PL/SQL modules zijn in 2.1 geen Modules meer, maar heten weer gewoon PL/SQL Definitions, terug te vinden onder de Server model tab. Wat helemaal wennen is, is het feit dat een Module niet omgezet kan worden in een PL/SQL Definition en vice versa. De language 'PL/SQL' is bij de module definitie domweg niet meer beschikbaar. Dit betekent dat als de Application Transformer bij het omzetten van een Business Function in een Module een verkeerde keuze maakt, dat dit dan niet meer gecorrigeerd kan worden. Ouderwets handwerk, het verwijderen en elders opnieuw opvoeren is het enige dat dan nog rest. De moraal hiervan is dat het nog belangrijker is geworden dan voorheen om een uitgebreide kwaliteitscheck te doen op de CRUD matrix en de response van de Business Functions.



Afbeelding 3 De Modules tab van de Design Editor

### Modules

Dit onderdeel van de navigatorree beslaat alles wat voorheen ook terug te vinden was onder de node 'Modules' in de navigatorree, met uitzondering van het beschrevene met betrekking tot PL/SQL objecten. Om de vroegere Module Structure Diagrammer overbodig te maken is er ook een node 'Module Network' opgenomen, waar de structuur van de modules hiërarchisch wordt weergegeven, vergelijkbaar met de hierarchy view in de RON (zie afbeelding 3).



Afbeelding 4 De DB Admin tab van de Design Editor

### DB Admin

Voor de zaken die door de ontwikkelaar-DBA worden vastgelegd in het ontwikkeltraject is het derde onderdeel onderkend: de DB Admin tab. Dit betreft met name zaken als autorisaties en sizing van tabellen en indexen. Het voordeel hiervan is dat de reguliere ontwikkelaars bijvoorbeeld de zogenaamde relationele eigenschappen van tabellen ('Relational Table Definition') kunnen vastleggen onder de tab 'Server Model' en dat ontwikkelaar-DBA's de zogenaamde implementatie eigenschappen ('Table Implementations') kunnen vastleggen onder tab 'DB Admin' (zie afbeelding 4). Deze scheiding geldt overigens niet alleen voor de invoer van de gegevens. Bij het genereren van tabellen kan gekozen worden voor een creatiescript met alleen de relationele eigenschappen of ook met de sizing eigenschappen. Om het principe van de implementatie-eigenschappen te kunnen ondersteunen, is het onderliggende datamodel van Designer gewijzigd met betrekking tot het vastleggen van definities van database objecten. Zo waren vroeger de sizinginformatie, en de verwijzing naar de tablespace waarin een object gecreëerd zou moeten worden, eigenschappen van dat object. Bij een tabel kon bijvoorbeeld maar één storage definition worden opgegeven, ook al zou die worden geïmplementeerd in zowel een (kleine) testdatabase als een (grote) produktiedatabase. De huidige structuur van Designer komt gelukkig meer overeen met de werkelijkheid. Onder de DB Admin tab zijn Databases vast te leggen, met behorend tot die databases de Users (lees: object owners), en behorend tot die Users de Schema Objects (lees: implementations). Op dit niveau, het implementation niveau, worden de 'dba-eigenschappen' van de databaseobjecten vastgelegd. Dit betekent dat een storage definition niet meer is gekoppeld aan de Table Definition, maar aan een Table Implementation, ofwel: aan een tabeldefinitie indien die gecreëerd gaat worden onder een bepaalde User in een bepaalde Database. Door deze wijziging in de Designerstructuur is het kinderspel geworden dezelfde tabeldefinitie met verschillende grootte in verschillende databases aan te maken.

Spijtig genoeg zijn de bedenkers van het implementatieconcept iets te ver doorgesloten. Ook voor het vastleggen van sequencedefinities is het implementatieniveau onderkend. Op het implementatieniveau dienen alle eigenschappen van de sequence, behalve de naam en het type, te worden vastgelegd.

In de praktijk zal het al snel zo zijn dat de eigenschappen van een willekeurige sequencedefinitie bij elke implementatie gelijk zijn. Het is ronduit hinderlijk dat bijvoorbeeld de start- en eindwaarde van de sequencedefinitie per implementatie vastgelegd dienen te worden.

### Distribution

Het vierde onderdeel van de Design Editor omvat alle objecten die met distributed databases te maken hebben. Hier kunnen zaken als Replication Groups, Snapshots en Database Links worden vastgelegd.

### Het Diagrammerwindow

Zoals bij de look-and-feel van de Design Editor al is aangestipt, is het met de huidige Designer erg gemakkelijk geworden een nieuw diagram te tekenen van een object uit de navigatortree. Met drag-and-drop kan er een 'diagram' worden gemaakt van het technisch datamodel (data schema diagram), van de moduleopbouw (module data usage diagram) en van de PL/SQL programmalogica (de logic editor). De opvolger van de vroegere preference-navigator is op vrijwel elk niveau in de navigatortree verborgen onder de rechter muisknop of op te roepen via het (Edit>Generator Preferences) menu. Een groot verschil met de vroegere navigator is het feit dat niet meer de complete hiërarchie van preferences wordt weergegeven, maar alleen het niveau waarop op dat moment de preferences worden onderhouden. De vroegere Module Structure Diagrammer is verdwenen uit Oracle Designer. In de navigatortree van zowel de RON als van de Design Editor is de structuur van modules duidelijk hiërarchisch weer te geven.

### De Forms Generator

Ten opzichte van de vorige release is de forms generator vele malen krachtiger geworden. Dit geldt overigens voor alle generatoren van Designer. Hierbij moet worden opgemerkt dat de toegenomen functionaliteit niet alleen aan de nieuwe generatoren is toe te schrijven, maar voor een minstens zo groot deel aan de meer gedetailleerde structuur van de informatie in de repository. Deze informatie kan daardoor vaak één-op-één door de generatoren worden overgenomen in de gegenereerde software. De belangrijkste aanpassing van de repository, waarvan de forms generator dankbaar gebruik maakt, is de toevoeging van twee nieuwe repository elementen. Dit zijn de herbruikbare module component en de applicatie logica. Deze uitbreidingen komen volledig op het conto van Oracle Designer en staan los van Oracle Developer.

De overige verbeteringen van de forms generator komen voort uit verbeteringen in Oracle Developer, die worden ondersteund vanuit Oracle Designer. Hiervan is het subclassingmechanisme al beschreven in het eerste deel (vorige Optimize) van dit artikel. Een andere verbetering zal dadelijk verder worden toegelicht. Het betreft de mogelijkheid blokken in Oracle Forms te baseren op procedures (!), in plaats van uitsluitend op tabellen.

### *Herbruikbare module componenten*

In de wat grotere informatiesystemen komt het vaak voor dat een bepaald blok in meerdere forms voorkomt met precies hetzelfde doel. Een voorbeeld hiervan is een query-only blok dat in meerdere forms als eerste blok wordt getoond om als context te fungeren. In deze forms zijn de (onderhoudbare) detailblokken telkens verschillend. Deze situatie is snel te herkennen: in het technisch datamodel is de tabel waarop het query-only blok is gebaseerd centraal gepositioneerd, als ware het 'een spin in het web'. De omliggende tabellen vormen allemaal details van de centrale tabel.

In grotere projecten zal het vaak het geval zijn dat de verwante forms door verschillende personen worden ontwikkeld. Het is dan wel wenselijk dat de gemeenschappelijke blokken dezelfde look-and-feel hebben, zoals layout, scherm prompts etc. Daarvoor is nu de herbruikbare module component in het leven geroepen. Een module component bestaat uit precies één base table usage met nul of meer lookup usages. Op het niveau van de module component kunnen preferences worden ingesteld. Door nu de module component in de repository te bestempelen als 'reusable', kan deze worden gebruikt in verschillende modules. De structuur van de module heeft ten opzichte van de vorige Designer release dus een extra niveau gekregen tussen de module en de table usage. Merk op dat dit niveau alleen in de ontwerpfase (in Oracle Designer) een rol speelt. In gegenereerde forms is onveranderd alleen het niveau 'blok' terug te vinden. De Repository API is ten opzichte van de vorige release van Designer uiteraard wel aangepast om de wijziging van de modulestructuur te ondersteunen. Dit impliceert dat ontwikkelproducten die gebruik maken van dit deel van de API zonder aanpassing niet meer correct zullen werken. Zo zullen niet alle Headstart Utilities van versie 3.4.x kunnen worden gebruikt tegen de nieuwe repository.

### *Applicatie logica*

Designer ontwikkelaars pur sang zullen met het beschikbaar komen van Designer 2.1 stellen dat 'Oracle Designer Oracle Developer bijna heeft opgeslokt'. Tot dusver was het grootste verschil tussen Designer en Developer inderdaad het feit dat je met Developer de code van elke trigger of program unit tot op de letter kon bepalen. Dat verschil is nu volledig verdwenen. In de repository is bij elke module applicatielogica vast te leggen. Afhankelijk van de soort module kan dit op meerdere niveaus. Zo kan voor Oracle Forms modules applicatielogica worden vastgelegd op module-, module component- en itemlevel. De taal

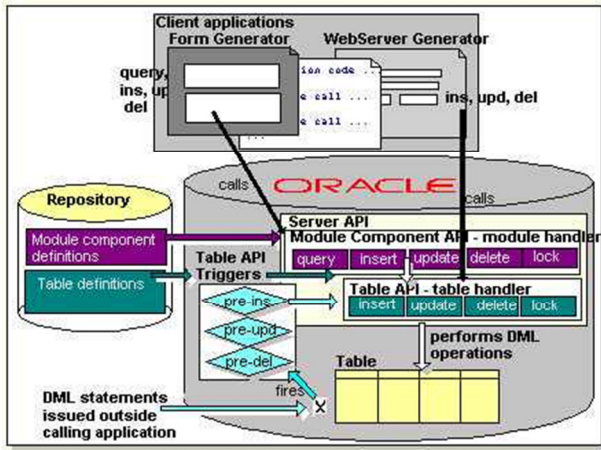
waarin de logica dient te worden gecodeerd is afhankelijk van de soort module. Bij forms is het bijvoorbeeld PL/SQL, maar bij web modules kan logica worden gecodeerd in javascript. De logica bestaat uit twee soorten: de events en de named routines. Afhankelijk van de soort module kan één of beide soorten worden vastgelegd. Bij Oracle Forms staan de events voor de triggers en de named routines voor program units van het form. Deze functionaliteit betekent dat we nu 100% generatie van modules kunnen bereiken, zonder gebruik te maken van een templatestructuur á la Headstart, Guidelines/2000, Design Assist etc. Om echter projectmatig te kunnen werken, met het kunnen doorvoeren van wijzigingen in applicatielogica op een hoger niveau dan de module, dus projectniveau (=applicatiesysteemniveau), zijn we nog steeds aangewezen op een dergelijke templatestructuur.

### *Procedureblokken*

Om het netwerkverkeer terug te dringen is Oracle Forms dusdanig aangepast, dat blokken niet langer uitsluitend op tabellen gebaseerd dienen te zijn. Het is nu ook mogelijk een blok te baseren op een procedure, of beter: op een aantal procedures. Het betreft dan een lock-, insert-, update-, delete- en select-procedure. Het voordeel hiervan is dat deze procedures als parameter een PL/SQL-tabel met meerdere records uit het blok mee kunnen krijgen. Ten opzichte van communicatie met de database in zuiver SQL verkleint deze manier het aantal zogenaamde 'network roundtrips' aanzienlijk. Uiteraard is Designer aangepast om deze functionaliteit te benutten. In de repository kan aangegeven worden dat een module component op een procedure is gebaseerd. De servergenerator kan vervolgens de benodigde procedures genereren (de zogenaamde TAPI en MAPI, zie de servergenerator).

### **De Library Generator**

Een nieuwe loot aan de Designer boom is de library generator. Hoewel de functionaliteit van deze generator weinig schokkend is, is hij toch onmisbaar om 100% genereren gestalte te kunnen geven. De generator doet niets meer en niets minder dan het één op één creëren van een fysieke PL/SQL library op basis van de moduledefinitie en applicatielogica van deze library in Designer. Uiteraard kan dit pad ook in omgekeerde richting bewandeld worden. Deze reverse engineering heet tegenwoordig 'Design Capture'. Eerlijkheid gebied te zeggen dat de parser van de Logic Editor (de editor waarmee je in Designer de applicatielogica vastlegt) nog lang niet het niveau heeft bereikt van de compiler in Oracle Developer. Veel programmeerfouten in applicatielogica zullen zich dus pas bij het genereren openbaren.



Afbeelding 5 Schematisch overzicht van de werking van de TAPI en de MAPI. Bron: Oracle

## De Server Generator

De server generator wordt steeds volwassen in de zin dat steeds meer databases worden ondersteund: Oracle8, Oracle7, Oracle Lite, Oracle Rdb, DB2/2, Microsoft SQL Server, Sybase, ANSI SQL DDL en ODBC databases. Daarnaast zijn er nu mogelijkheden voor generatie van objecten voor symmetrische replicatie. De belangrijkste verbetering is echter de mogelijkheid van het genereren van een Table API (TAPI) en een Module Component API (MAPI). Deze api's hebben een tweeledige functie:

- het aanroepbaar zijn vanuit formsblokken (ze vormen dan de procedures waar een formsblok op gebaseerd kan worden).
- het herbergen van validaties en acties die aan de serverkant uitgevoerd dienen te worden.

In afbeelding 5 wordt een schematisch overzicht gegeven van de werking van de api's. Rechtsonder is de tabel te zien waarop het gewenst is bewerkingen uit te voeren. Deze worden uitgevoerd door de eerste schil, de zogenaamde table handler of TAPI. Deze kan direct worden aangeroepen vanuit gegenereerde Webmodules. De TAPI kan op zijn beurt ook worden aangeroepen door de tweede schil, de zogenaamde module handler of MAPI. De procedures van de MAPI kunnen vervolgens worden aangeroepen door op procedures gebaseerde formsblokken.

Omdat voor webmodules altijd al table handlers benodigd waren, is het concept van de twee verschillende handlers toegepast. De TAPI was er al (voorheen gegenereerd door de Web Server Generator), dus waarom niet hergebruikt? Om er voor te zorgen dat alle validaties en serveracties ook worden uitgevoerd als de tabel niet via de api's wordt bewerkt maar direct met DML statements buiten de clientapplicaties om, kunnen er ook table api triggers worden gegenereerd. Deze triggers zorgen er alsnog voor dat de juiste api-procedures worden uitgevoerd. In de gegenereerde code van de api's en api triggers kan ook eigen code worden toegevoegd. Deze wordt als applicatielogica in de repository vastgelegd (zie de navigatortree in afbeelding 2, 'Table API/Trigger Logic'). Hierdoor is het niet meer nodig zelf databasetriggers te

definiëren of de source ervan door Headstart Utilities te laten genereren. De api's bevatten default veel functionaliteit en eventueel toevoegen van applicatielogica in de repository volstaat.

## Het ontwikkelproces

Wat betekenen de wijzigingen in Designer nu voor het ontwikkelproces? Voor het eerst kan nu de DBA ook verplicht worden gesteld gegevens vast te leggen in Designer. Tot dusver konden de dba's hun ei niet echt kwijt in designer, bijvoorbeeld doordat er slechts de sizing kon worden vastgelegd die geschikt was voor één doelomgeving. Zodoende schreef elke dba z'n eigen scripts voor het genereren van sizinginformatie in de DDL creatiescripts, voor het berekenen van de benodigde schijfruimte voor de objecten en voor het genereren van grants en synoniemen. De uitdaging voor de dba's wordt nu het 'omschrijven' van deze scripts zodat deze zich gaan baseren op de Repository API en de vastlegging van bron én resultaten van de scripts in de repository plaats gaat vinden. Voor de ontwikkelaar geldt uiteraard dat hij efficiënt gebruik kan maken van alle nieuwe features van Designer 2.1. Zodra echter gebruik wordt gemaakt van de TAPI en de MAPI, zal een deel van de software die zich traditioneel (lees: in Designer 1.3.2) uitsluitend op de client bevond, zich naar de server gaan verplaatsen. Een treffend voorbeeld hiervan is een op een TAPI/MAPI gebaseerd form dat wordt uitgebreid met een extra item. Het volstaat nu niet meer om uitsluitend de moduledefinitie van het form aan te passen en het form opnieuw te genereren. Na het aanpassen van de moduledefinitie moet eerst de MAPI opnieuw worden gegenereerd en gecreëerd in de ontwikkelomgeving. Hoewel dit een klein en voor de hand liggend verschil lijkt vergeleken met de vroegere werkwijze, betekent het wel dat het wijzigen van forms minder vrijblijvend is geworden. Bij promotie van het form van de ontwikkel- naar de test- of productieomgeving moet nu immers ook een script worden opgeleverd om de database in lijn te krijgen met de nieuwe TAPI en/of MAPI. Het versiebeheer -toch al niet het sterkste punt van Oracle Designer- is er dus niet eenvoudiger op geworden, integendeel.

## De toekomst van Oracle Designer

Voor IT-managers zijn het barre tijden. Het moordend tempo waarmee Oracle de laatste tijd major releases van Oracle Designer uitbrengt is voor velen beangstigend. Niet alleen is het voor bedrijven qua kennis van de producten amper bij te houden, ook het moeten uitrollen van 'weer een nieuwe versie' zorgt in menige organisatie voor enige 'upgrademoehheid'. Met de nieuwe Oracle Designer 6.0 in zicht (binnenkort beschikbaar) is het wellicht verstandig een eventuele overstap van 1.3.2 naar 2.1 uit te stellen en ineens de sprong te nemen (wagen?) naar Designer 6.0. De belangrijkste reden hiervoor is dat de door de gebruikersorganisatie gebruikte forms en reportsversie wederom een major upgrade moet

ondergaan. Designer 1.3.2 genereert immers Forms 4.5, Designer 2.1 genereert Forms 5.0 en Designer 6.0 zal Forms 6.0 genereren.

Wat kunnen we van Designer 6.0 zoal verwachten? Zoals het er nu naar uitziet wordt het een zogenaamde maintenance-release. Geen noemenswaardige functionaliteitswijzigingen dus, maar vooral bugfixes. Wel zullen waarschijnlijk JavaBeans als userinterface-componenten in Developer 6.0 worden ondersteund vanuit Designer 6.0.

De grote wijzigingen in Designer worden uitgesteld tot versie 6.5. Deze versie komt als bèta beschikbaar medio mei en wordt produktie tegen de herfst. Dit zal de eerste versie zijn waarin grootschalige ondersteuning van Java terug te vinden is. Het is in Oracle8i mogelijk stored procedures, stored functions en database triggers in Java te programmeren. Deze zullen met Designer 6.5 vastgelegd en gegenereerd kunnen worden. De andere belangrijke verbetering is versiebeheer op objectniveau. Designer 6.5 maakt daarvoor gebruik van een 'externe repository', de dan tot zelfstandig produkt uitgegroeide Oracle Repository 7.0. Het jaar wordt afgesloten met het uitkomen van Designer 6.6. Deze versie bevat een UML (Unified Modeling Language) modelleringstool vergelijkbaar met de bij Designer 2.1 nog als apart produkt meegeleverde Object Database Designer. Ook wordt er een transformatie mogelijk gemaakt vanuit de UML diagrammen naar de Design Editor. Het releaseschema van Oracle in ogeschouw nemend wordt het allesbehalve saai de komende tijd. En dat is een behoorlijk understatement.

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

(c) Copyright 1999 A New Link bv