# Quest Code Tester for Oracle

## *Automatisch unittesten van PL/SQL modules*

***Dit artikel is geschreven naar aanleiding van een onderzoek naar de bruikbaarheid van Quest Code Tester for Oracle binnen de Oracle Fusion ontwikkelstraat van Capgemini. Aan de hand van twee voorbeelden wordt het gebruik van Quest Code Tester for Oracle toegelicht.***

In een moderne ontwikkelstraat is het inrichten van automatische unittesten onontbeerlijk. Daar waar ontwikkelomgevingen als Java en .Net wel voorzien in automatische unittesttools, zoals JUnit respectievelijk NUnit, waren er tot op heden nog maar weinig tools beschikbaar die de PL/SQL-ontwikkelaar ondersteunen in het definiëren en uitvoeren van gestructureerde, automatische unittesten. Steven Feuerstein onderkende dit probleem al enkele jaren geleden en heeft zich de afgelopen tijd sterk gemaakt voor het ontwikkelen van een tool voor het automatisch unittesten van PL/SQL modules. Uiteindelijk heeft dit geresulteerd in een nieuw product in de Quest portfolio: Quest Code Tester for Oracle.

### Quest Code Tester for Oracle

Momenteel zijn er twee versies van Quest Code Tester for Oracle beschikbaar, een freeware en een commerciële versie. De freeware versie is een iets oudere versie (versie 1.2.5) dan de meest recente commerciële versie. Voor dit artikel is gebruik gemaakt van de meest recent beschikbare commerciële versie (versie 1.5.1).

Quest Code Tester for Oracle (hierna te noemen QCTO) bestaat uit vier hoofdcomponenten:

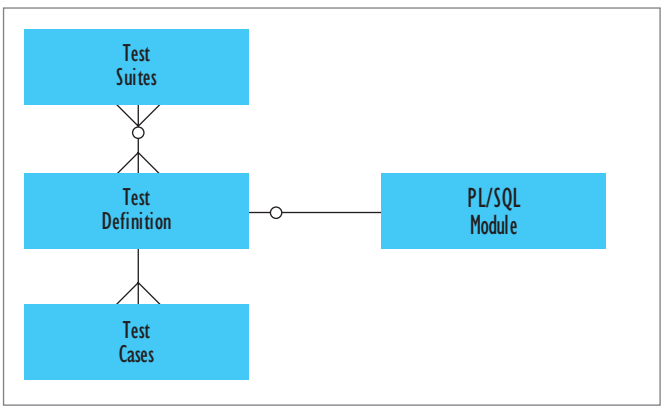
1. Test Dashboard (Figuur 2). Dit is het hoofdscherm en biedt per Oracle schema een overzicht van de gedefinieerde testdefinities en testsuites.
2. Test Builder (Figuur 3). Voor het vastleggen van testdefinities via een grafische user interface.
3. Test Viewer (Figuur 6). Voor het in detail bekijken van de uitkomsten van de verschillende testruns.
4. Test Reporter (Figuur 7). Voor een overzicht van alle bestaande testdefinities en de uitkomst van de verschillende testruns.

### Installatie QCTO

QCTO maakt gebruik van een repository welke in een bestaand of in een nieuw databaseschema geïnstalleerd kan worden. Uit het oogpunt van beheersbaarheid is er voor gekozen om QCTO in een apart schema zonder public synonymen te installeren. Op deze manier wordt er een duidelijke scheiding gemaakt tussen de applicatie en de unittest omgeving. De repository owner moet dan natuurlijk wel rechten krijgen op de objecten uit het te testen schema. Dit moet vanuit de applicatie zelf geregeld worden. De installatie verloopt via een grafische interface, welke je stap voor stap door het installatieproces leidt. Er moeten enkele simpele vragen worden beantwoord, voordat de installatie verder automatisch verloopt.

### Opzet unittesten binnen QCTO

Binnen QCTO wordt per PL/SQL module één testdefinitie vastgelegd. Binnen een testdefinitie worden één of meerdere testgevallen opgevoerd, dit zijn testcases.



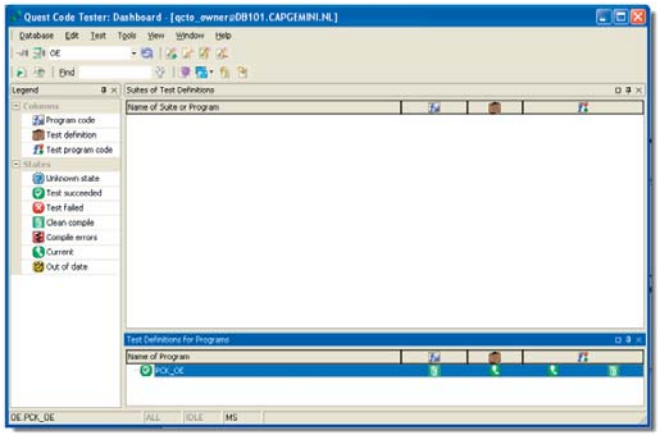

*Figuur 1. Opzet unittesten binnen QCTO*

Testdefinities kunnen gegroepeerd worden in testsuites (Figuur 1). Via de aanroep van één testsuite is het mogelijk om meerde-

re unittesten uit te voeren. Zo zou er bijvoorbeeld een testsuite voor de hele applicatie vastgelegd kunnen worden, en een testsuite per release of per deelapplicatie. Afhankelijk van welke testsuite er aangeroepen wordt, worden dan alle unittesten uitgevoerd, of een subset van de unittesten.

## Test Dashboard

Als QCTO wordt gestart en er is ingelogd met de QCTO repository owner, wordt het Test Dashboard (Figuur 2) getoond. Binnen dit scherm wordt per databaseschema de reeds gedefinieerde testsuites en de gedefinieerde testdefinities getoond.




*Figuur 2. Test Dashboard*

Vanuit dit scherm kunnen testdefinities opgevoerd, gewijzigd en uitgevoerd worden.

## Test Builder

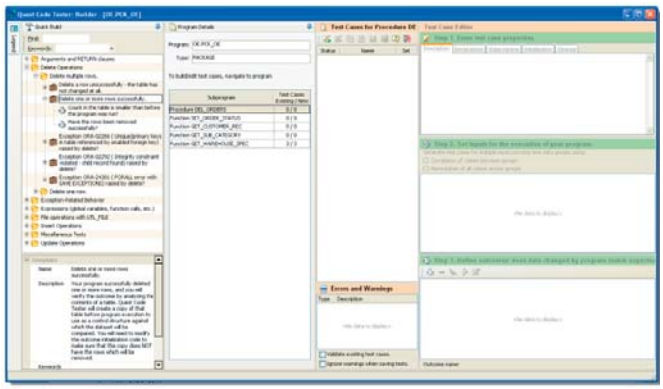
Via Test Builder (Figuur 3) kunnen bestaande testcases worden gewijzigd en kunnen er nieuwe testcases aan een bestaande testdefinitie worden toegevoegd. Om de kracht en eenvoud van het opvoeren van testcases te tonen, is er een package PCK_OE ontwikkeld onder het demoschema OE. Het testen van scalar typen (uitvoerparameters die één waarde kunnen bevatten, zoals number, varchar2, date) is de makkelijkste vorm van uitvoer van een PL/SQL module, en kan zondermeer via QCTO worden getest. Veel PL/SQL modules zullen complexer van aard zijn en zullen data in database tabellen raken via DML statements. Om dit soort modules te testen zal dus eerst een referentiedataset klaargezet moeten worden waarmee de verwachte uitkomst vergeleken kan worden. Vervolgens wordt de module uitgevoerd en zal daarna de aangepaste dataset gecontroleerd moeten worden tegen de referentiedataset. Alle testen binnen QCTO zijn onafhankelijk van elkaar en beginnen en eindigen altijd met een ROLLBACK. Het is dus niet mogelijk om meerdere testdefinities in een bepaalde volgorde uit te voeren, om zo de samenhang tussen modules te testen Het testen van de samenhang tussen verschillende modules vindt namelijk plaats tijdens de systeemtest en niet tijdens de unittest.

De testen voor DML modules zijn met QCTO eenvoudig te definiëren. De package PCK_OE bevat een procedure DEL_ORDERS (Listing 1) welke alle orders voor een bepaald jaar verwijdert. Het jaar moet minimaal zes jaar of ouder zijn.

```
--
-- Delete orders from a specific year
--
PROCEDURE del_orders (p_year in number) IS
  c_current_year constant number(4) := to_number(to_char (sysdate,
'yyyy'));
  c_keep_term constant number(2) := 5; -- Keep orders at least for <n>
years
BEGIN
  --
  -- Only 6 years and older
  --
  if p_year between c_current_year - c_keep_term and c_current_year
  then
    raise_application_error (-20001, 'Year must be smaller than ' ||
to_char(c_current_year - c_keep_term));
  end if;
  --
  -- Delete order_item
  --
  delete from order_items
  where order_id in
  (select order_id
   from orders
   where to_number (to_char (order_date, 'yyyy')) = p_year
  )
  ;
  --
  if sql%rowcount = 0
  then
    raise no_data_found;
  end if;
  --
  -- Delete orders
  --
  delete from orders
  where to_number (to_char (order_date, 'yyyy')) = p_year
  ;
END del_orders;
```

*Listing 1. Procedure del_orders*

Test Builder is opgedeeld in drie delen. Aan de linkerkant Quick Build, waar meer dan 60 voorgedefinieerde standaard testcases als template worden meegeleverd. In het midden de Program Details waar alle procedures en functions binnen een package en het aantal reeds gedefinieerde testcases worden getoond. Hier wordt gekozen voor de procedure DEL_ORDERS. De rechterkant is voor het toevoegen en wijzigen van testcases. De template 'Delete one or more rows successfully' wordt naar de rechterkant gesleept. Vervolgens moet eerst aangege-

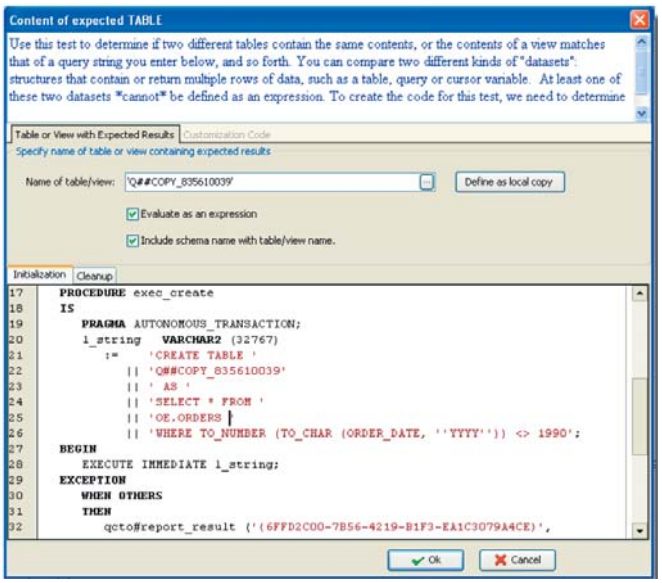


*Figuur 3. Test Builder*

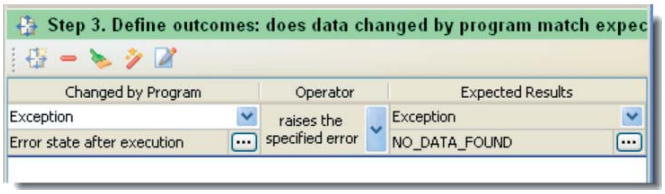
ven worden uit welke tabel records verwijderd gaan worden, in dit geval OE.ORDERS. Nadat QCTO het raamwerk van de testcases heeft aangemaakt, moeten eerst de invoerparameters worden ingegeven. In dit geval krijgt P_YEAR de waarde 1990.

Aangezien de inhoud van een tabel wordt gewijzigd, is het verwachte resultaat van de test iets complexer. Echter via QCTO is dit relatief eenvoudig te definiëren. Via het scherm uit Figuur 4 wordt eerst de tabel OE.ORDERS als naam van de tabel ingevuld en vervolgens wordt via de button [Define as local copy] zowel 'Initialization' alsmede 'Cleanup' code gegenereerd.

Vervolgens moet nog de initialisatie code worden aangepast, om alleen die records aan te maken die ook in de resultaatset verwacht worden. In Figuur 4 is alleen de WHERE-clause handmatig toegevoegd. De testcase 'Count in the table is smaller than before the program was run' kan verwijderd worden. Om de testdefinitie compleet te maken, moet ook voor de foutsitu-




*Figuur 4. Content of expected table*



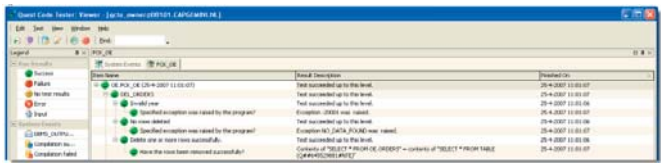

*Figuur 5. Testen PL/SQL exception*

aties een testcase worden vastgelegd, in dit geval een exception NO_DATA_FOUND (Figuur 5) en een exception -20001.

Als de testdefinitie compleet is, wordt door QCTO een PL/SQL package gegenereerd met testcode voor het uitvoeren van de daadwerkelijke test. Na het uitvoeren van de testdefinitie zijn de resultaten te bekijken in de Test Viewer (Figuur 6).

## Test Viewer

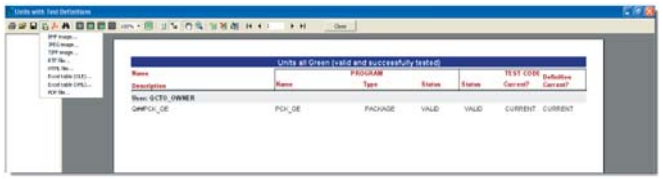
Via Test Viewer kunnen in detail de resultaten van de testrun bekeken worden. Er kan ingezoomd worden op testcase niveau, en er wordt per testcase aangegeven of het resultaat wel of niet aan de verwachting voldoet.



*Figuur 6. Testrun procedure del_orders*

## Test Reporter

Via Test Reporter zijn verschillende overzichten te genereren van de PL/SQL modules met en zonder testdefinities en de uitkomsten van de verschillende testruns.



*Figuur 7. Test Reporter*

## Boolean expressions

Een belangrijk type tests zijn Boolean expressions. Via een boolean expression is het namelijk mogelijk om Oracle types te testen welke (nog) niet door QCTO worden ondersteund. Zo biedt de huidige versie van QCTO nog geen volledige ondersteuning voor XML types. Via een boolean expression zijn XML-types wel eenvoudig te testen. Om twee XML-types te vergelijken wordt gebruik gemaakt van de package DBI_XML_COMPARE. Deze open source package is te downloaden via

http://www.db-innovations.co.uk/products_5.htm. Via de functie is_identical_xml kan er getest worden of twee XML types aan elkaar gelijk zijn.

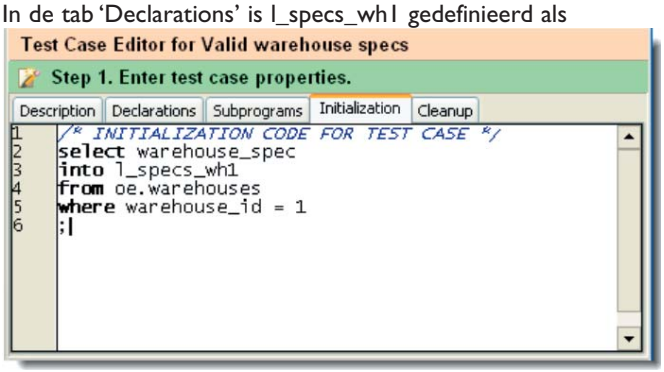
In de package PCK_OE zit een functie GET_WAREHOUSE_SPEC, welke voor een warehouse de bijbehorende specificaties in XML formaat ophaalt.

```
--
-- Get warehouse specification
--
FUNCTION get_warehouse_spec (p_warehouse_id in warehouses.warehouse_
id%type) RETURN XMLTYPE IS
    l_warehouse_spec warehouses.warehouse_spec%type;
BEGIN
   select warehouse_spec
   into l_warehouse_spec
   from warehouses
   where warehouse_id = p_warehouse_id
   ;
   --
   return (l_warehouse_spec);
END get_warehouse_spec;
```

*Listing 2. Function get_warehouse_spec*

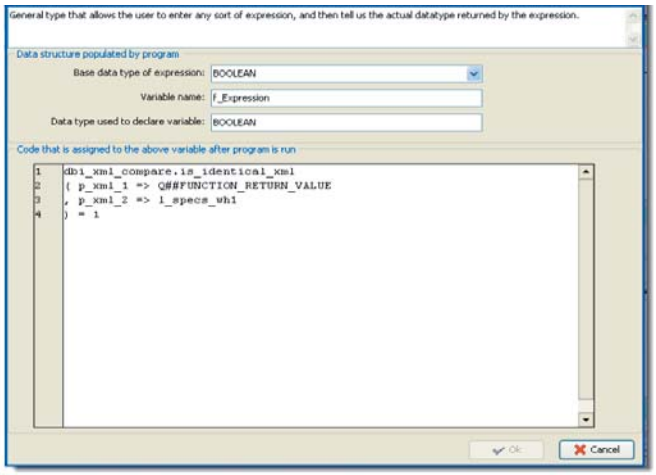
Via Test Builder wordt aan de bestaande testdefinitie van PCK_OE een testcase toegevoegd voor de functie GET_WAREHOUSE_SPEC. Allereerst wordt de initialisatie code vastgelegd zoals getoond in Figuur 8.

In de tab 'Declarations' is l_specs_wh1 gedefinieerd als




*Figuur 8. Testcase initialization*

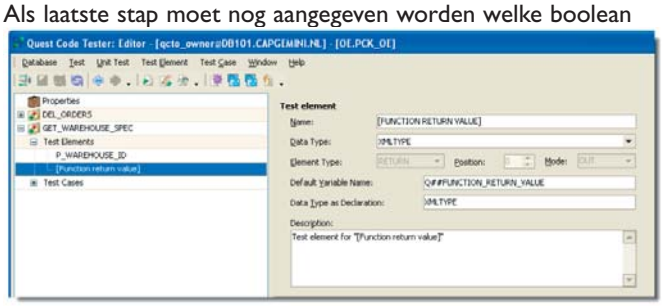
XMLTYPE. De code in de tab 'Initialization' wordt aan het begin van de testcase uitgevoerd en kan gebruikt worden om specifieke zaken van te voren klaar te zetten.
De code in de tab 'Cleanup' wordt na het uitvoeren van een testcase uitgevoerd en kan gebruikt worden om specifieke zaken weer op te ruimen. Na het invoeren van de invoerparameter (P_WAREHOUSE_ID krijgt de waarde 1), wordt als verwacht eindresultaat gekozen voor het type uitkomst 'Expression' en moeten de properties voor de expression ingevuld worden via het scherm uit Figuur 9).




*Figuur 9. Expression properties*

Er wordt gekozen voor een expression van het type BOOLEAN en vervolgens wordt handmatig de code toegevoegd die leidt tot een boolean expression. In dit geval de aanroep van de functie DBI_XML_COMPARE.IS_IDENTICAL_XML met de bijbehorende variabelen. De naam van de uitvoervariabele kan via de Test Editor (Figuur 10) gevonden worden. Via de Test Editor kan in detail de testdefinitie en de bijbehorende testcases bekeken en eventueel gewijzigd worden. Tevens kan de source van de gegenereerde testpackage bekeken worden en kunnen eventuele compilatiefouten opgespoord worden.

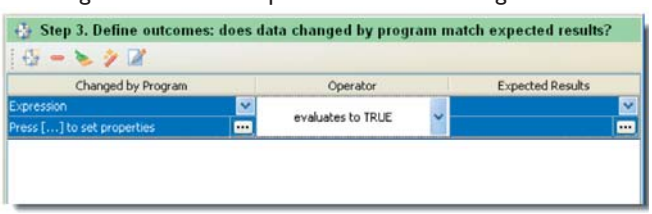
Als laatste stap moet nog aangegeven worden welke boolean




*Figuur 10. Test Editor*

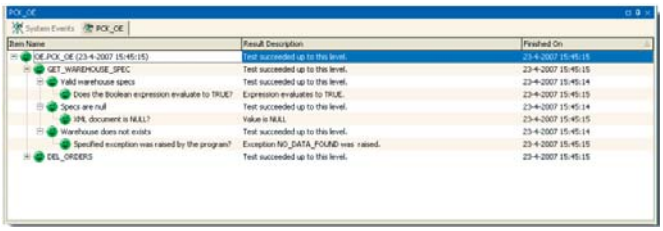
uitkomst er verwacht wordt, in dit geval TRUE (Figuur 11).

Na het genereren en compileren kan de test uitgevoerd wor-




*Figuur 11. Expression evaluates to TRUE*

den en kan wederom via de Test Viewer de resultaten bekeken worden (Figuur 12).



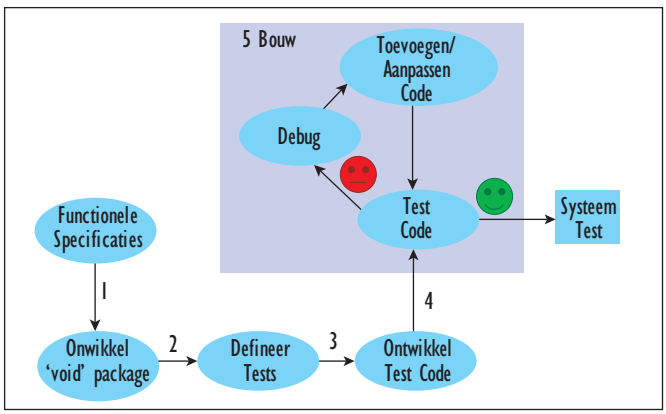
*Figuur 12. Testrun function get_warehouse_spec*

## Quest Code Tester Community

Via de Quest Code Tester Community (http://unittest.inside.quest.com) is veel informatie te vinden over het gebruik van QCTO. Tevens zijn er meerdere forums om bevindingen te loggen en vragen te stellen. Ook is hier de zogenaamde Helper Package te downloaden. In de Helper Package zit handige functionaliteit welke (nog) niet standaard in QCTO voor handen is.

## Test Driven Development

Door het inzetten van automatische unittesttools is het mogelijk om PL/SQL modules te gaan ontwikkelen volgens de Test Driven Development (TDD) methode (Figuur 13). TDD gaat er vanuit dat een ontwikkelaar eerst zijn testdefinities maakt op basis van de functionele specificaties, voordat er ontwikkeld mag gaan worden. In PL/SQL land betekent dit dus dat eerst de package specification wordt ontwikkeld met de bijbehorende package body zonder logica (void package). Als de package compileert, kunnen vervolgens de testdefinitie met bijbehorende testcases in QCTO gedefinieerd en uitgevoerd worden. Zolang alle testcases goed gaan, is het niet nodig om logica aan de package toe te voegen. Op het moment dat er een of meerdere testcases fout gaan, wordt de minimaal benodigde code aan de package toegevoegd om er voor te zorgen dat de testcases wel goed gaan (Het zou natuurlijk ook kunnen dat er nog een fout zit in de testcase). Dit proces wordt net zolang doorlopen totdat alle testcases die volgens de specificaties nodig zijn, goed gaan en de package gereed is. Om dit proces goed te kunnen doorlopen is het van belang dat de ontwikkelaar snel




*Figuur 13. Test Driven Development*

zijn testdefinities kan vastleggen en vervolgens deze testen snel en herhaalbaar kan uitvoeren. Met de komst van QCTO is dit nu mogelijk.

## Conclusie

Via QCTO is een PL/SQL ontwikkelaar in staat om snel en declaratief met een minimale programmeerinspanning testdefinities vast te leggen. De huidige versie van QCTO heeft één serieus probleem, en dat is dat de performance indien QCTO via een netwerk naar een database moet, zeer te wensen overlaat. Dit is verholpen door een Oracle database te installeren op dezelfde Windows server als QCTO. Dit probleem is bij Quest bekend, en is hopelijk in een nieuwe versie (1.6) opgelost. Voor de rest zitten er enkele kleine onvolkomenheden in, waarvoor meestal een workaround te vinden is. Hopelijk zijn de meeste hiervan in versie 1.5.2 opgelost, welke waarschijnlijk op het moment dat dit artikel verschenen is, beschikbaar is.

QCTO is een echte aanvulling op de bestaande toolset voor de PL/SQL ontwikkelaar en kan bijdragen aan het vergroten van de professionaliteit van de PL/SQL gemeenschap. QCTO maakt onderdeel uit van de Oracle Fusion ontwikkelstraat binnen het Accelerated Delivery Center (ADC) van Capgemini.

**Sjoerd Aalbers** is werkzaam als Senior Consultant bij Capgemini (e-mail: sjoerd.aalbers@capgemini.com).