PL/SQL Unit Tests mit SQL Developer

Perry Pakull
Trivadis AG
Zürich

Schlüsselworte
Unit Test, SQL Developer, PL/SQL

Einleitung

Unit Tests sind zwar ein aufwendiger aber auch notwendiger Bestandteil eines modernen Software Entwicklungsprozesses. Im Oracle Datenbankumfeld sind die vorhandenen Frameworks für PL/SQL Unit Tests überschaubar. Der lizenzkostenfreie SQL Developer von Oracle bietet eine Vielzahl von Assistenten und Funktionen für den Entwickler, die den Einstieg in Unit Tests für PL/SQL Programme erleichtern. Der Vortrag beschreibt anhand von Beispielen die Möglichkeiten des SQL Developers für die Erstellung, Verwaltung und Automatisierung von Unit Tests.

Unit Test Repository erstellen

SQL Developer benötigt ein Repository für Unit Test Funktionen. Das Repository besteht aus Tabellen, Views und anderen Datenbankobjekten, die typischerweise in ein eigenes Schema der Oracle Datenbank installiert werden. Um ein neues Repository zu installieren sind drei Schritte erforderlich, die über die SQL Developer Benutzeroberfläche auszuführen sind.
Zuerst wird ein neuer Datenbankbenutzer für das Repository angelegt. Das ist über eine Verbindung mit DBA-Berechtigungen möglich. Ein Rechts-Klick auf den Knoten "Other Users" öffnet ein Kontextmenü in dem der Befehl "Create User" gewählt werden kann. SQL Developer empfiehlt UNIT_TEST_REPOS als Benutzernamen. Die weiteren Pflichtfelder wie Passwort und Tablespace Informationen sind ebenfalls auszufüllen. Bei den System Privilegien ist das Recht "CREATE SESSION" zu vergeben.
Das Anlegen einer neuen Verbindung für den erstellten Datenbankbenutzer ist der zweite Schritt. Der Dialog in Abbildung 1ist im Menü über die Auswahl "Tools, Unit Test, Manage Users" erreichbar.

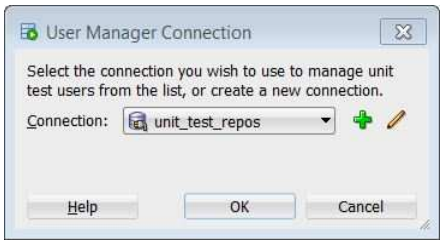

Abb. 1: Dialog für eine neue Verbindung zum Unit Test Repository

Hier kann eine neue Verbindung für den Datenbankbenutzer UNIT_TEST_REPOS erstellt werden. Danach erscheinen weitere Dialogfenster und Abfragen, um die erforderlichen Berechtigungen für die Administration des Unit Test Repositories zu vergeben. UNIT_TEST_REPOS wird damit als Administrator berechtigt und eingetragen.
Das Unit Test Repository wird über den Menüpunkt "Tools, Unit Test, Select Current Repository" erstellt. Hier wird die erstellte Verbindung für den Datenbankbenutzer angegeben. Da in diesem Schema noch kein Repository vorhanden ist, kann ein neues angelegt werden. Funktionen für die Administration des Repositories sind in einem eigenen Menü über die Auswahl "Tools, Unit Test" verfügbar.

Die Berechtigungen für das Unit Test Repository sind über zwei Datenbank Rollen geregelt, die bei der Erstellung des Repositories angelegt werden. Die Rolle UT_REPO_USER berechtigt einen Datenbankbenutzer Unit Tests zu erstellen und auszuführen. Diese Rolle kann von einem Repository Administrator vergeben werden. Administratoren besitzen die Rolle UT_REPO_ADMINISTRATOR. Nach der Erstellung des Repositories sind die Datenbankbenutzer SYS und UNIT_TEST_REPOS Administratoren.
Sobald ein Repository erstellt wurde kann man ein bestehendes Datenbankschema entsprechend berechtigen, um Unit Tests zu erstellen. Die Auswahl "Tools, Unit Test, Manage Users" verlangt eine Verbindung mit Administrator Berechtigungen für das Repository. Mit dem Dialog in Abbildung 2 kann nun ein bestehendes Datenbankschema ausgewählt werden.

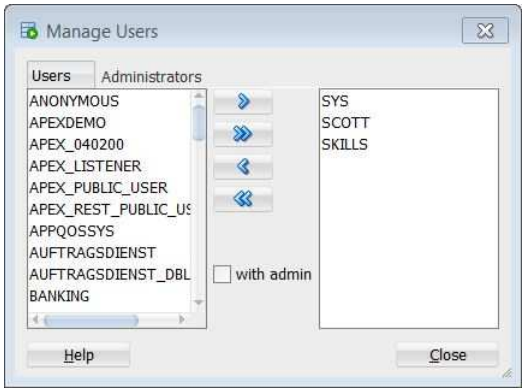

Abb. 2: Manage Repository Users

Das Repository enthält die einzelnen Unit Tests in Form von Metadaten. Mehrere Unit Tests können in "Suites" gruppiert werden. Für die Anzeige von Testergebnissen gibt es fertige Auswertungen im Repository. Daten für Eingabeparameter sowie Vor- und Nachbereitungsprozesse für Unit Tests sind als wiederverwendbare Komponenten in einer Library gespeichert. Das Navigationsfenster in Abbildung 3 bietet eine Sicht auf das Repository und die gespeicherten Inhalte. Das Fenster kann über den Menüpunkt "View, Unit Tests" geöffnet werden.

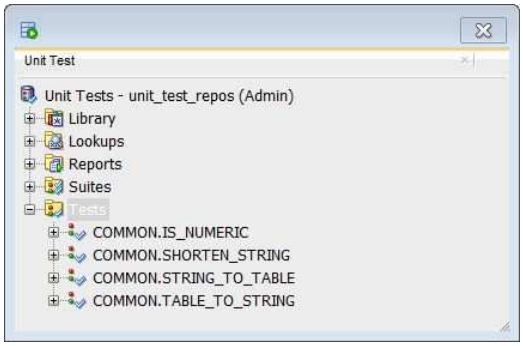

Abb. 3: Unit Test Repository

Unit Tests erstellen und ausführen
Die Erstellung eines Unit Tests ist sehr intuitiv. Ein Rechts-Klick auf eine Prozedur, Funktion oder Methode eines Packages im Navigator einer SQL Developer Verbindung öffnet ein Kontextmenü wie in Abbildung 4 zu sehen und der Befehl zur Erstellung eines Unit Tests ist sichtbar. Ein Unit Test kann auch über einen Rechts-Klick auf den Knoten Tests im Navigationsfenster für Unit Tests erstellt werden.

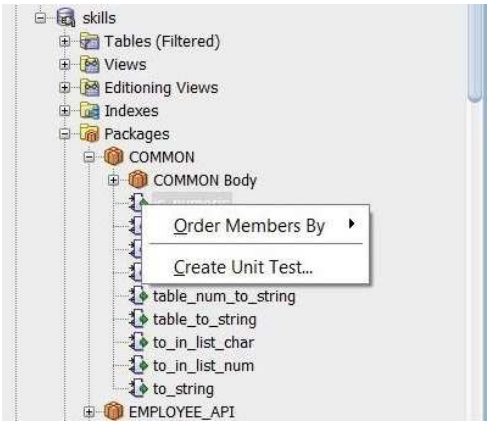

Abb. 4: Kontextmenü einer Package Methode im Navigator

Der aufgerufene Assistent führt über 6 Schritte zur Erstellung eines Unit Tests. Im ersten Schritt wird der Test Name von der entsprechenden PL/SQL Einheit übernommen. Zusätzlich kann die Erstellung von Test Implementierungen beeinflusst werden. Für einen Test wird standardmäßig nur eine Implementierung angelegt.
Der zweite Schritt ermöglicht die Definition von Startup Prozessen, die immer dann erforderlich sind, wenn ein Unit Test Datenbankobjekte oder Daten in einer bestimmten Konstellation benötigt oder Daten durch Transaktionen verändert. SQL Developer unterscheidet zwei wählbare Prozessarten. Die Prozessart "Table or Row Copy" kopiert Daten aus einer Quelltabelle in eine Zieltabelle, so dass der ursprüngliche Datenzustand nach dem Test wieder hergestellt werden kann. Die definierte Zieltabelle wird neu angelegt bzw. überschrieben falls sie schon existiert. Mit der zweiten Prozessart "User PL/SQL Code" kann der Startup Prozess frei programmiert werden.
Der nächste Dialog ermöglicht die Eingabe von Werten für die ermittelten Parameter der PL/SQL Einheit. Bei Funktionen kann der Return-Wert als Ergebnis angegeben werden. Das angegebene Ergebnis wird zur Laufzeit mit dem Return-Wert verglichen. Stimmen die Werte nicht überein, ist der Test fehlgeschlagen. Anstelle der manuellen Eingabe können die Werte auch dynamisch mit einer Query übergeben werden. Das spezielle Format der "Dynamic Value Query" ist im nachfolgenden Listing zu sehen.

```
select ? as RETURNS$, ? as P_EMP_ID from ? where ?
```

Der Alias liefert die Information, wie die Werte auf die Parameter zu verteilen sind. Für Funktionen ist auch der Return-Wert anzugeben. Liefert die Query mehrere Datensätze, wird die Test Implementierung entsprechend oft ausgeführt. Eine "Dynamic Value Query" kann im Repository hinterlegt und so für verschiedene Tests verwendet werden. Das erwartete Gesamtergebnis des Unit Tests ist entweder erfolgreich ("Success") oder fehlerhaft ("Exception"). Dadurch sind sowohl Positiv- als auch Negativ-Tests definierbar.
Für die Überprüfung der Testergebnisse im vierten Schritt stehen wiederum vorgegebene Prozesse zur Verfügung, die das Vergleichen von Daten unterstützen und vereinfachen. Auch hier gibt es die Prozessart "User PL/SQL Code" mit der eine Überprüfung frei programmiert werden kann.
Das Gegenstück zu den Startup Prozessen sind Teardown Prozesse im fünften Schritt. Hier können Datenbankobjekte und Daten aufgeräumt werden, die mit Startup Prozessen erstellt wurden.
Die finale Zusammenfassung der erfassten Definitionen bildet den Abschluss des Assistenten, um den Unit Test zu speichern oder zu verwerfen. Der Anwender kann jederzeit auf die vorhergehenden Schritte zurück navigieren.

Der erstellte Unit Test ist jetzt im Unit Test Navigator sichtbar. Ein Klick auf den Namen und der Unit Test wird mit allen Details geöffnet. Das Kontextmenü wird über einen Rechts-Klick geöffnet und bietet die Funktionen Löschen, Umbenennen, Kopieren, Implementierung ergänzen, Ausführen, Testergebnisse löschen, Synchronisation und Export an. Alle Bestandteile des Tests können im Unit Test Editor, der in Abbildung 5 zu sehen ist, nachträglich geändert oder ergänzt werden.

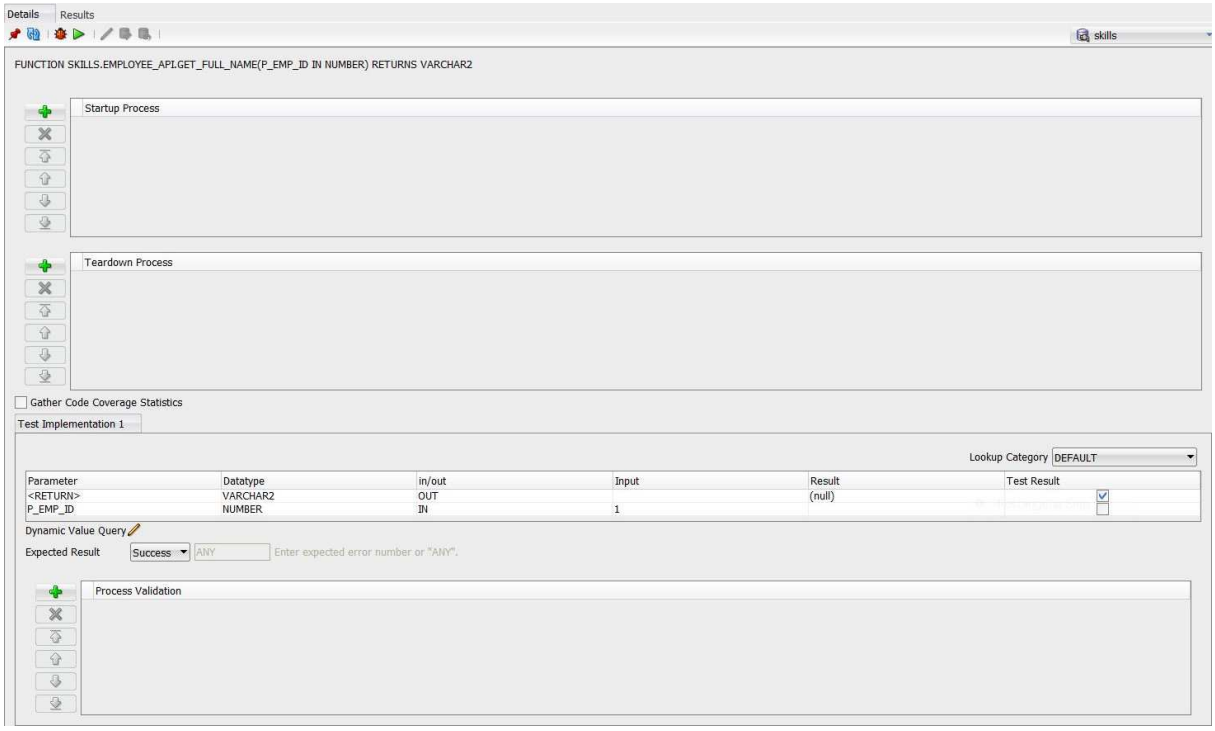

Abb. 5: Unit Test Editor

Ein Unit Test wird über die Funktionstaste F9 synchron ausgeführt und das Ergebnis wie in Abbildung 6 angezeigt.

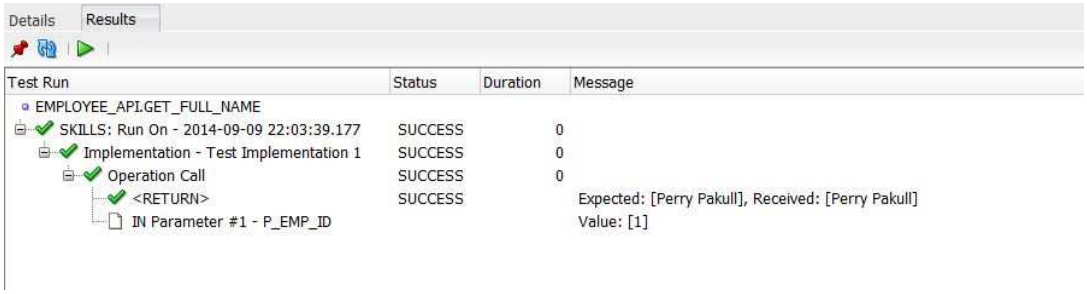

Abb. 6: Unit Test Ergebnis

Best Practice

Für den Einstieg und das Verständnis der SQL Developer Unit Test Funktionen ist die Lektüre der SQL Developer Hilfe empfehlenswert. Darin enthalten sind ein vollständiges Tutorial und Empfehlungen für die Verwendung der vielfältigen Objekte und Funktionen.

Für den erfolgreichen Einsatz von Unit Tests ist eine Strategie für die Vorgehensweise und die Granularität der Tests erforderlich. Ein Unit Test adressiert die kleinste testbare Einheit einer Software. In der Datenbank sind das einzelne Prozeduren und Funktionen oder Prozeduren und

Funktionen in einem Package oder Object Type Body. Für neue Applikationen kann parallel zur Entwicklung des Source Codes auch die Entwicklung der Unit Tests erfolgen. Für bestehende Applikationen ist zunächst die Einteilung der PL/SQL Objekte in funktionale Gruppen sinnvoll. Eine Gruppe kann als Suite abgebildet werden. Schrittweise können dann einzelne Unit Tests den Suites zugeordnet werden.
Der Name für einen Unit Test ist auf 120 Zeichen begrenzt und muss eindeutig sein. Eine sinnvolle Benennung ist daher wichtig. Bei der Erstellung eines Tests wird der Name des Datenbankobjektes vorgeschlagen. Bei Prozeduren und Funktionen aus einem Package wird der Package Name vorangestellt. Diese vorgeschlagene Form der Benennung ist durchaus sinnvoll und garantiert eine einfache Zuordnung der Tests zu den Datenbankobjekten. Falls mehr als ein Test für ein Datenbankobjekt erforderlich ist oder unterschiedliche Versionen gehalten werden, so sollte die vorgeschlagene Nummerierung durch eine sinnvolle Erweiterung im Namen angepasst werden. Wenn Datenbankobjekte mit dem gleichen Namen aus unterschiedlichen Schemas vorhanden sind, dann sollte der Schema Name ergänzt werden.
Die Benennung der Test Implementierungen ist an den Inhalt des Tests anzupassen. Die vorgeschlagene Nummerierung der Implementierungen ist wenig aussagekräftig. Jeder Test sollte eine Implementierung mit den Standard-Eingabewerten enthalten. Weitere Implementierungen mit NULL-Werten, Minimum- und Maximum-Werten ist ebenfalls zu empfehlen.
Unbedingt empfehlenswert ist die Verwendung von Test Suites für eine Gruppierung der Tests. Die Unit Tests zu einem Package können in einer Suite mit dem Package Namen gruppiert werden. Suites sollten hierarchisch geordnet werden, so dass alle Tests durch den Aufruf einer Master-Suite ausgeführt werden können.
Die Unit Test Library im Repository gibt die Möglichkeit die Komponenten von Unit Tests zu modularisieren und mehrfach zu verwenden. Die Objekte können aus der Library kopiert oder vererbt werden.
Die automatisierte Ausführung der Tests mit dem Command-Line Interface ist sehr empfehlenswert. Die Auswertung der Testergebnisse kann durch Abfragen der Ergebnis-Tabellen im Repository ebenfalls automatisiert werden.
Das Unit Test Repository sollte in ein eigenes, separates Datenbankschema installiert werden. Ein Repository pro Datenbank ist ausreichend. Bei Bedarf können auch mehrere installiert werden.

Automatisierung von Unit Tests
Software Entwicklungsprozesse setzen heute auf Continuous Integration und Continuous Delivery, um die Qualität der erstellten Software zu gewährleisten und zu verbessern. Die Automatisierung von Tests ist dabei ein zentraler Bestandteil. Das Command-Line Interface des SQL Developers bietet die Möglichkeit, einen Unit Test automatisch auszuführen. Die Syntax für den Aufruf eines Unit Tests ist im folgenden Listing beschrieben.

```
sdcli64 unittest
        -run
        -test | -suite
        -id <id> |-name <name>
        -repo <connection name>
        -db <connection name>
       {-return <return_id>}
       {-log <1,2,3>}
```

Die Parameter "repo" und "db" benötigen jeweils SQL Developer Verbindungen, einmal für das Unit Test Repository und für das Schema in dem die Unit Tests auszuführen sind. Ersetzt man den Parameter "test" durch den Parameter "suite" wird eine ganze Test Suite ausgeführt. Da eine Test Suite auch andere Test Suites enthalten kann ist der Aufruf aller vorhandenen Unit Tests bei entsprechender Organisation der Tests möglich.

Das Ergebnis des Unit Tests ist im SQL Developer Unit Test Repository über die Reports "All Test Runs" und "All Suite Runs" einsehbar. Für eine automatisierte Auswertung der Testergebnisse kann der optionale Command-Line Parameter "return" beim Aufruf angegeben werden. Die Testergebnisse werden dann unter dieser ID im Unit Test Repository abgelegt und können direkt über die Datenbank ausgewertet werden. Die Repository Tabellen UT_TEST_RESULTS, UT_TEST_IMPL_RESULTS, UT_TEST_IMPL_ARG_RESULTS, UT_TEST_IMPL_VAL_RESULTS, UT_SUITE_RESULTS und UT_SUITE_ITEM_RESULTS enthalten die Testergebnisse.

Export und Import
Der Export von Unit Tests, Suites und anderen Objekten aus dem Repository ist eine gute Hilfe für die Automatisierung von Unit Tests. Der Export der Objekte erfolgt in eine XML Datei und kann in das Repository einer anderen Datenbank importiert werden. Durch einen Rechts-Klick auf den Namen eines Unit Tests im Unit Test Navigator wird ein Kontextmenü geöffnet und die Funktion "Export to File" ist aufrufbar. Markiert man mehrere Unit Tests im Navigator so werden alle Objekte in die XML Datei exportiert. Der Export enthält die markierten Objekte und alle abhängigen Objekte. Beim Export einer Suite werden alle zugeordneten Unit Tests exportiert.
Beim Import werden alle Objekte der Datei im Repository neu angelegt. Sind Objekte mit gleichen Namen bereits vorhanden, so kann der Benutzer wählen, ob diese überschrieben oder ausgelassen werden sollen. Der Importdialog in Abbildung 7 kann im Menü über die Auswahl "Tools, Unit Test, Import from File" aufgerufen werden.

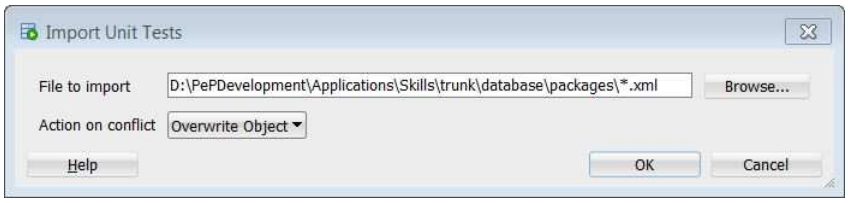

Abb. 7: Import Unit Tests

Export und Import stehen ebenfalls im Command-Line Interface zur Verfügung. Die Syntax für den Export von Repository Objekten ist im folgenden Listing zu sehen.

```
unittest -exp
         -test | suite | validation | startup | teardown | query
         -id <id> | -name <name>
         -repo <connection name>
         -file <file name>
```

Die Syntax im nachfolgenden Listing für den Import ist vergleichsweise einfach. Alle Objekte in der angegebenen Datei werden in das angegebene Repository importiert.

```
unittest -imp
         -repo <connection name>
         -file <file name>
        [-overwrite | -skip]
```

Die exportierten XML Dateien bieten auch eine gute Möglichkeit, die Unit Tests zu sichern und über eine Versionskontrolle zu versionieren. Änderungen am PL/SQL Source Code bedeuten meistens auch Änderungen an den dazugehörigen Unit Tests.

Fazit
Der Gesamteindruck der SQL Developer Unit Test Funktionen ist durchweg positiv. Datenbank Entwickler, die ihre PL/SQL Objekte mit Unit Tests testen wollen oder müssen, verfügen mit dem

SQL Developer über eine gute, integrierte Umgebung für die Entwicklung des Source Codes und der Tests.
Die Unit Test Funktionen sind schlüssig und vernünftig implementiert, daher ist die Verwendung an vielen Stellen einfach und intuitiv. Die Erstellung der Startup und Teardown Prozesse wird durch sinnvolle Vorlagen ergänzt, aber nicht eingeschränkt. Die Command-Line Interface Funktionen sind hilfreich für eine Automatisierung.
Zu den Nachteilen gehört die verfügbare Lookup-Funktionalität bei der Erstellung der Test Implementierungen. Die gewünschte Kategorie kann nicht aus einer Liste gewählt, sondern nur umständlich in den Einstellungen vorgegeben werden. Validierungsprozesse können nicht auf benutzerdefinierte Datentypen zugreifen. Der Zugriff auf die Werte ist zwar möglich, aber nur über den Datentyp VARCHAR2. Das Repository Navigationsfenster bietet keine Möglichkeit nach Tests oder anderen Objekten zu suchen. Bei vielen Objekten geht die Übersicht schnell verloren.

Kontaktadresse

Perry Pakull
Trivadis AG
Europa-Strasse 5
CH-8152 Glattbrugg

Telefon: +41-58-459 55 55
Fax: +41-58-459 55 95
E-Mail: perry.pakull@trivadis.com
Internet: www.trivadis.com