SQLcl Quo vadis SQL*Plus? Das neue SQL*Plus in der Praxis

Gunther Pippèrr
GPI Consult
München

Schlüsselworte

SQL, SQL*Plus, SQLcl, Scripting

Einleitung

Ist es nun soweit? Gehört nun auch SQL*Plus bald der Geschichte an?

Mit SQLcl stellt das Team hinter dem SQL Developer ein Kommando Zeilenwerkzeug zur Verfügung, das durchaus das Potential hat, SQL*Plus in der Entwicklung abzulösen.

Welche Features erwarten uns und ist das aber wirklich ein echter Ersatz für SQL*Plus?

Im Vortrag wird eine Übersicht über die Funktionalität von SQLcl gegeben und auf die Vor-und Nachteile gegenüber SQL*Plus eingegangen.

Der Schwerpunkt des Vortrags liegt in den Scripting Möglichkeiten des Java Scripting Interfaces in SQL*Plus.

Aufgezeigt wird die Verwendung der "eingebauten" JavaScript Funktionalität und wie mit „Jython " in Python als Skript Sprache unter SQLcl gearbeitet werden kann.

In übersichtlichen Beispielen werden Einsatzmöglichkeiten in der Praxis dargestellt.

Übersicht

SQLcl ist in Java implementiert und ist auch Teil des SQL Developers.

Über http://www.oracle.com/technetwork/developer-tools/sql-developer/downloads/index.html kann ganz unten auf der Webseite die Command Line - SQLcl - Early Adopter Release geladen werden, aktuell in der Version vom 16.September.2016.

Ziel des SQLcl Projektes ist es neben der Kompatibilität zu SQL*Plus auch eine Reihe von neuen Features anzubieten

Installation

Eine eigentliche Installation ist nicht notwendig, es reicht die Zip Datei nach dem Download in ein Verzeichnis zu entpacken. Eine Java 8 Installation wird allerdings zwingend vorausgesetzt.

Nach dem Entpacken die „sql.bat“ unter Windows bzw “sql“ unter Linux starten und hoffen, dass es klappt, treten Java Fehler auf, die Dateien anpassen, die Skripte sind nicht sonderlich robust umgesetzt.

Erste Schritte

Die meisten Oracle SQL*Plus Format Befehlt wurde implementiert, leider aber nicht 100% aller Anweisungen. Das ist zum Teil etwas lästig, da ältere SQL*Plus Script nicht 1zu1 in beiden Umgebungen lauffähig sind.

Hier ist es angeraten über das Forum https://community.oracle.com/community/database/developer-tools/sql_developer/sqlcl mit der Entwicklung Kontakt aufzunehmen und nachzufragen, evtl. wird doch noch das eine oder andere Feature implementiert.

Neue Funktionen gegenüber SQL*Plus

Das wohl interessante Feature ist der Befehlszeilen Buffer unter Linux, das haben wohl schon Generation von DBA's vermisst .-) .

Unter Linux liegt die History unter ~/.sqlcl/history.xml, hier liegt auch in der Datei aliases.xml die Alias Definition.

Allerdings sind, wie früher die „login.sql", auch diese Dateien nicht wirklich geschützt. Hier ergeben sich wieder gute Angriffsvektoren um Kollegen ungewollten Code „unterzuschieben". Hier hätte Oracle etwas sorgsamer arbeiten können, Oracle lernen das anscheinend nie an so etwas zu denken.

Einen Alias definieren

```
SQL> alias ls=SELECT object_name from USER_OBJECTS;
SQL> ls

OBJECT_NAME
------------------------------------------------------------
RECORDS_ID_SEQUENCE
```

Oracle Fehlernummern auflösen

```
SQL> oerr ora 600

00600. 00000 -  "internal error code, arguments: [%s], [%s], [%s], [%s], [%s], [%s], [%s], [%s], [%s], [%s], [%s], [%s]"
*Cause:    This is the generic internal error number for Oracle program
           exceptions. It indicates that a process has encountered a low-level,
           unexpected condition. The first argument is the internal message
           number. This argument and the database version number are critical in
           identifying the root cause and the potential impact to your system.
```

Mit SQL „hint" verschiedene Ausgabenformate definieren

```
SELECT /*csv*/       * FROM employees;          -- Comma-separated values
SELECT /*delimited*/ * FROM employees;          -- (same as csv)
SELECT /*fixed*/     * FROM employees;          -- Fixed-width fields with trailing blanks
SELECT /*html*/      * FROM employees;          -- Marked-up HTML table
SELECT /*insert*/    * FROM employees;          -- SQL INSERT statements
SELECT /*loader*/    * FROM employees;          -- Pipe-delimited format suitable for SQL*Loader
SELECT /*text*/      * FROM employees;          -- Plain text
SELECT /*xml*/       * FROM employees;          -- Tagged XML
```

DDL erstellen

```
SQL> ddl domains
  CREATE TABLE "PDNS"."DOMAINS"
   (    "ID" NUMBER(*,0) NOT NULL ENABLE,
        "NAME" VARCHAR2(255) NOT NULL ENABLE,
        "MASTER" VARCHAR2(128) DEFAULT NULL,
        "LAST_CHECK" NUMBER(*,0) DEFAULT NULL,
        "TYPE" VARCHAR2(6) NOT NULL ENABLE,
        "NOTIFIED_SERIAL" NUMBER(10,0) DEFAULT NULL,
        "ACCOUNT" VARCHAR2(40) DEFAULT NULL,
```

Scripting mit SQLcl

Das interessanteste und wichtigste Feature ist das Scripting mit SQLcl.

Leider ist aber das Ganze noch nicht so wirklich dokumentiert.

Einige Beispiele finden sich hier ⇒ https://github.com/oracle/oracle-db-tools/tree/master/sqlcl

Folgende Objekte können in Script (default Java Script Syntax) direkt angesprochen werden:

- sqlcl - SQLCL selbst wie sqlcl.setStmt(„select * from dual“) und sqlcl.run() um das dann auszuführen
- ctx - Object vom Typ ScriptContext, kann direkt angesprochen werden - wie ctx.write(„String“)
- util - wie var user=util.executeReturnOneCol('select user from dual');

Hier das erste Hello World Beispiel:

```
script
ctx.write("Hello Word\n");
/
```

```
SQL> script
  2  ctx.write("Hello Word\n");
  3  /
Hello Word
```

Soll aber auf bestimmte Java Klassen zugegriffen werden, müssen diese zuvor eingebunden werden wie zum Beispiel ⇒ „var System = Java.type(„java.lang.System“);“

Beispiel - SQLcl Oberflächen Sprache per Script in der login.sql setzen:

```
Script
  var System  = Java.type("java.lang.System");
  System.setProperty("user.lang","en");
  System.setProperty("user.country","us");

  System.out.println( System.getProperty("user.lang"));
  System.out.println( System.getProperty("user.country"));

/
```

Formatangaben anpassen über die FormatRegistry:

```
script
  var FormatRegistry = Java.type("oracle.dbtools.raptor.format.FormatRegistry");
  FormatRegistry.setLineTerminator("-\t");
/
```

Script Sprachen wie Jython einbinden

Es können auch verschiedene Script Sprachen eingesetzt werden, je nach persönlichen Wunsch.

Möglich wird das durch die Implementierung des Java Scripting Interfaces JSR-223.

Um zum Beispiel Jython einzubinden muss nur die Jar Datei in den Klassenpfad beim Aufruf von SQLcl aufgenommen werden.

1. Download der Jar Datei jython-standalone-2.7.0.jar von http://www.jython.org/downloads.html
2. Kopieren nach $SQLCL_HOME/lib
3. Einbinden in den Klassen Pfad in der $SQLCL_HOME/bin/sql.bat bzw. sql
   - ```
     53
     54 set CPFILE=%CPFILE%;%SQL_HOME%\lib\jython-standalone-2.7.0.jar
     55
     ```
   - Linux: export CLASSPATH=$CLASSPATH:/opt/oracle/products/sqlcl/lib/jython-standalone-2.7.0.jar
4. Erstellen des ersten jython scripts helloWorld.py
   - ```
     print "Hello World!"
     ```

Aufruf mit „SQL> script helloWorld.py“ => „Hello World“

```
SQL> script helloWorld.py
Hello World
```

D.h. die ScriptEngine erkennt über die Datei Endung, was für ein Interpreter verwandt werden soll!

Leider führt der erste Test unter Windows gleich zu einem „console: Failed to install ": java.nio.charset.UnsupportedCharsetException: cp65001.“

Das lässt sich zwar über den Java Start Parameter „-Dpython.console.encoding=UTF-8“ bekämpfen, zeigt aber wieder auf, dass hier noch viel Arbeit für Oracle notwendig ist um das Konsolenwerkzeug unter Windows wirklich lauffähig zu bekommen.

Fazit

SQLcl ist auf den richtigen Weg ein interessantes zusätzliches Werkzeug zu werden und kann beim Deployment und im täglichen Einsatz bei der Entwicklung so einiges an Vorteilen aufzuzeigen.

Leider ist die Software bei weiten noch nicht so stabil wie SQL*Plus und das wichtige Feature. mit NICHT UTF-8 Zeichensätzen unter Windows umgehen zu können, ist noch nicht zu 100% umgesetzt. Es ist zu vermuten, dass dies zwar ein generelles Java Problem ist, nicht desto trotz ist das aber entscheidend für ein Kommandowerkzeug unter Windows! Folgt man den Foren Einträgen, fällt sehr unangenehm auf, dass die Entwicklung anscheinend unter MAC oder Linux stattfindet und die ganzen Zeichensatz und UTF-8 Probleme nur sehr zögerlich angenommen werden.

Das Einbinden des Java Scripting Interfaces JSR-223 ermöglicht es, auch andere Skriptsprachen zu verwenden und damit sehr flexible Lösungen zu entwickeln. Das funktioniert auch von Release zu Release besser, ein deutliches Manko ist aber die sehr dünne Dokumentation. Es wird zwar auf einige Beispiel verwiesen, das Ganze hat aber noch einen sehr prototypenhaften Charakter.

Trotz dieser Nachteile lohnt sich der Einsatz des Tools, viele kleine hilfreiche Features helfen im Alltag und durch die 100% Java Lösung lässt sich das Ganze auch einfach mit der eigenen Software für das Deployment ausliefen (vorausgesetzt eine Java 8 Umgebung steht zur Verfügung und Windows ist nicht die Hauptplattform!).

Kontaktadresse:

Gunther Pippèrr

GPI Consult

Schwanthalerstr. 82

D-80336 München

Telefon:	+49 (0)89 53 026 418
Mobil:	+49(0)171 8065113
E-Mail	gunther@pipperr.de
Internet:	http://www.pipperr.de/dokuwiki/doku.php