ESPRESSO TUTORIALS

Jörg Böke

Schnelleinstieg in SQLScript für SAP HANA®

- Einführung in SAP HANA SQLScript
- Anwendungsbeispiele von SQLScript in Prozeduren, Funktionen und BW/4HANA
- SQLScript in ABAP Core Data Services (CDS) für Reportingzwecke
- Tipps und Tricks für den performanten Einsatz von SQLScript

Inhaltsverzeichnis

2 Anwendungsbeispiele in HANA

In diesem Kapitel werde ich auf die Verwendung der Funktionalitäten von SQLScript in den verschiedenen Bereichen eines SAP-Systems eingehen.

Die Nutzung von Funktionen und Prozeduren ermöglicht es Ihnen, Ihren Code lesbar zu halten, indem Sie die verschiedenen Aufgaben, die ausgeführt werden sollen, nicht in einer einzigen Prozedur mit beispielsweise 500 Zeilen Script erstellen, sondern diese in mehrere Prozeduren und Funktionen aufteilen.

Eine Übersicht der HANA-SQLScript-Referenz erhalten Sie unter:

https://help.sap.com/viewer/4fe29514fd584807ac9f2a04f6754767/2.0.05/en-US/20ff532c751910148657c32fe3431a9f.html

Im Folgenden werden Sie zusätzlich zu den Funktionen, die die SAP mit Auslieferung der HANA DB bereitstellt, eigene Funktionen und Prozeduren anlegen. Diese Erweiterung des SQLScript-Standards ermöglicht Ihnen, die Funktionalität immer weiter auszubauen und Funktionen genau auf Ihre individuellen Bedürfnisse anzupassen.

2.1 HANA-SQLScript-Funktionen

Funktionen (Functions) dienen in HANA SQLScript der einmaligen Definition einer immer wiederkehrenden Funktionalität, um diese dann an anderer Stelle ebenfalls zu nutzen. Falls später eine Anpassung am Funktionscode notwendig wird, muss nur eine zentrale Funktion anstatt einer großen Anzahl von Stellen geändert werden. Dadurch kann der Code wiederverwendbar und lesbar gehalten werden. Die SAP liefert in SQLScript eine Vielzahl an Funktionen in Abhängigkeit von der jeweiligen HANA-DB-Version aus.

Nachfolgend liste ich einige der *SAP HANA Functions* auf, die Sie später noch nutzen werden bzw. schon in Abschnitt 1.14.1 kennengelernt haben.

- UCASE (identisch mit UPPER-Funktion): Der Befehl dient dazu, alles in Großbuchstaben umzusetzen.
- UPPER: Der Befehl dient dazu, alles in Großbuchstaben umzusetzen.
- RIGHT: Dieser Befehl gibt ab dem rechten String-Ende nach links eine Anzahl von Zeichen aus.
- LEFT: Dieser Befehl gibt vom linken String-Anfang nach rechts eine Anzahl von Zeichen aus.

Für Funktionen besteht in HANA 1.0 keine Möglichkeit, rekursiv eine Hierarchie der Tabelleninhalte (z. B. Kunden) aufzubauen; ab der Version HANA 2.0 kann dies durch SAP umgesetzt werden. Es lohnt sich daher, die Funktionsmöglichkeiten der jeweiligen Versionen von HANA SQLScript kennenzulernen, um das eigene Wissen auf dem aktuellen Stand zu halten.

In den folgenden Beispielen möchte ich Ihnen zeigen, wie Sie eigene Funktionen erstellen und nutzen können. Um SQLScript übersichtlich zu halten, ist es sehr sinnvoll, Teilaufgaben in entsprechende Funktionen auszulagern.

2.1.1 SQLScript-Funktionsdeklaration

SQLScript-Funktionen bestehen aus einem Kopf, der definiert, in welchem Schema die Funktion erzeugt werden soll, dem Namen der Funktion sowie den Input-Parametern.

Bei Funktionen wird der Rückgabewert über RETURNS übergeben. Dieser kann je nach Deklaration eine einzelne Variable oder Tabelleninhalt sein.

Im nächsten Teil, der standardmäßig deklariert werden sollte (siehe Listing 2.1), ist die Sprache (in diesem Fall `SQLScript`) definiert sowie, mit welcher Berechtigung `SQL SECURITY INVOKER` (Developer, Invoker – also Entwickler oder zum Zeitpunkt der Ausführung gültige Anwender/User) die Funktion ausgeführt wird.

Im weiteren Verlauf kann mit dem Code `READS SQL DATA AS` (Prozedur darf nur lesenden Zugriff auf die DB haben) oder `AS` (Funktion darf auch Daten verändern) definiert werden, ob eine Prozedur intern lesend oder schreibend arbeitet.

Ab der Zeile BEGIN folgt das von Ihnen manuell erstellte Coding wie der Aufruf von Funktionen, SELECT Statements etc. Mit END wird die Funktion anschließend beendet.

Jeglicher Kommentar sollte mit zwei Minuszeichen `--` beginnen, damit dieser nicht als SQLScript interpretiert werden kann.

```
--Funktion Rumpfprogramm
CREATE FUNCTION MEIN_SCHEMA.GET_MY_NAME (
          IN ip_MEIN_INPUT NVARCHAR(50) ,
          )
      RETURNS op_MEINOUTPUT NVARCHAR(256)
      LANGUAGE sqlscript
      SQL SECURITY INVOKER
      READ SQL DATA AS – oder nur AS

BEGIN
--Hier steht Ihr Coding

END;
```

Listing 2.1: Beispiel für den Funktionskopf/Deklaration

2.1.2 Nested Functions

Eine *Nested Function* ist eine Funktion, die intern eine andere Funktion aufruft. Wie bereits in der Einleitung beschrieben, kann dies dazu

dienen, SQLScript lesbarer zu halten, da z. B. anstelle von 100 Zeilen nur zwei bis drei Zeilen benötigt werden.

Anhand der Berechnung der Anzahl von Tagen zwischen zwei Daten wird dies im folgenden Beispiel veranschaulicht (siehe Listing 2.2).

Als Erstes müssen Sie die Funktion erstellen, die Sie später in den Routinen benutzen wollen.

```
--Nested function für Aufruf in Prozedur
CREATE FUNCTION get_days(
IN first_d date, IN second_d date )

RETURNS num_days bigint
AS
BEGIN
      num_days =  days_between (:first_d,:second_d  );
END;
```

Listing 2.2: Beispiel für den Aufruf von Nested Functions

Wie zu erkennen ist, nutzt diese Funktion wiederum eine SAP-Funktion (DAYS_BETWEEN), wodurch auch eine mehrstufige Schachtelung ermöglicht wird.

2.1.3 Verwendung von User Defined Functions (UDF)

Im Gegensatz zur ABAP-Programmierung, bei der Funktionen nur im Coding genutzt werden können, verwenden Sie in SQLScript SAP-Funktionen oder Ihre eigenen Funktionen (UDF) direkt mit SQL-Kommandos.

Stellen Sie sich vor, Sie möchten aus der Tabelle KUNDE den Namen sowie Vornamen auslesen und beide mit einem bestimmten Trennzeichen (z. B. --) ausgeben. Die gewünschte Ausgabe sollte also wie folgt sein: (Nachname--Vorname) »Böke--Jörg«.

Sie können dies direkt als SQLScript-Abfrage implementieren oder in eine Funktion auslagern, sollte die Anforderung häufiger benötigt werden. Ein kleines Beispiel dazu zeigt Listing 2.3.

In der Funktion, die aus drei Input-Parametern und einem Return-Parameter besteht, wird überprüft, ob innerhalb der IF-Anweisung ein Trennzeichen geliefert wird. Je nach Ergebnis der Anweisung wird dieses mittels PIPE (II)-Funktionalität verkettet und ausgegeben.

```
--Beispiel einer Namens-Funktion
CREATE FUNCTION GET_FULL_NAME (
          IN ip_vorname NVARCHAR(50) ,
          IN ip_trennzeichen NVARCHAR(50) ,
          IN ip_nachname NVARCHAR(50)
          )
                  RETURNS op_fullname NVARCHAR(256)
      LANGUAGE sqlscript
      SQL SECURITY INVOKER
    AS
      BEGIN
          IF :ip_trennzeichen IS NULL THEN
                  op_fullname = :ip_nachname || '** ' ||
:ip_vorname;
          ELSE
                  op_fullname = :ip_nachname || :ip_
trennzeichen ||    :ip_vorname ;
          END IF;
END;
```

Listing 2.3: Beispiel für eine User Defined Function

Diese Funktion können Sie nun mittels einer SQL-Konsole testen.

```
select GET_FULL_NAME('Jörg','--','Böke') from dummy
```

Bei Übergabe eines Trennzeichens erhalten Sie das in Abbildung 2.1 gezeigte Ergebnis.

B Index