

Tipps & Tricks: UTL_FILE II

Bereich:	PL/SQL	Erstellung:	06/2002 HA
Versionsinfo:	9.2, 10.2, 11.1	Letzte Überarbeitung:	06/2009 EF

 [Als PDF Downloaden!](#)

Bearbeitung von ASCII- und Binärdateien mit UTL_FILE

Ascii- und binäre Dateien bearbeiten mit UTL_FILE

Mit dem Package UTL_FILE können seit Version 7.3 Dateien gelesen und geschrieben werden. Mit Version 9.2 wurden einige nützliche Prozeduren/Funktionen ergänzt, und das Konzept der Zugriffsberechtigungen wurde auf Verzeichnisse umgestellt.

Informationen zu den wichtigsten Funktionen UTL_FILE.READ UND - WRITE finden Sie [hier](#).

Zugriffsberechtigungen

Bei Oracle-Versionen vor 9.2 muss in der Datei init<SID>.ora der Parameter UTL_FILE_DIR gesetzt werden (dafür muss man die Datenbank herunterfahren und neu starten). Es können keine expliziten Rechte an einzelne User vergeben werden.

Ab Version 9.2 wird auch bei UTL_FILE, analog zu BFILES, mit DIRECTORY-Objekten gearbeitet, auf die explizit Rechte vergeben werden können (Bis zur Version 9.2.0.7 funktionierten die Rechte nicht wirklich so, wie sie eigentlich sollten (GRANT READ ON DIRECTORY bewirkte, dass der Grantee sowohl lesen als auch schreiben(!) konnte, GRANT WRITE ON DIRECTORY liess hingegen keinen Schreibzugriff zu)

Beispiel:

```
CREATE DIRECTORY for_utl_file AS 'C:\temp';
GRANT READ, write ON DIRECTORY for_utl_file TO scott;
GRANT EXECUTE ON UTL_FILE TO scott;
```

Hier wird ein kleines Textfile in C:\temp angelegt

```
DECLARE
    v_file UTL_FILE.FILE_TYPE;
BEGIN
    v_file := UTL_FILE.FOPEN('FOR_UTL_FILE', 'test.txt', 'w');
    UTL_FILE.PUT_LINE(v_file, 'Hello World!');
    UTL_FILE.PUT_LINE(v_file, 'Grüße aus Unterhaching');
    UTL_FILE.PUT_LINE(v_file, 'von MuniQSoft');
    UTL_FILE.FCLOSE(v_file);
END;
/
```

Bereits mit Version 9.0.x wurde parallel zu den vorhandenen ASCII-Routinen UNICODE-Unterstützung eingeführt mit FOPEN_NCHAR, GET_LINE_NCHAR, PUT_NCHAR und PUT_LINE_NCHAR.

Ab Version 9.2 ist es auch möglich, über UTL_FILE

- mittels GET_RAW, PUT_RAW Binärdateien zu lesen und zu schreiben
- mittels FGETPOS und FSEEK die Position innerhalb einer geöffneten Datei abzufragen und zu verändern
- mittels FCOPY, FREMOVE, FRENAME eine Datei ganz oder teilweise zu kopieren, zu löschen oder umzubenennen (bzw. zu verschieben):
- mittels FGETATTR Attribute einer Datei auszulesen

GET_RAW und PUT_RAW wurden in der Form getestet, dass eine Datei (verschiedene Bildformate und ein Word-Dokument) unter Windows XP und unter Unix ausgelesen und anschließend der Inhalt in eine neue Datei geschrieben wurde.

Unter Unix wurden die Testdateien (ein Bild und eine Word-Datei) via ftp auf den Rechner geschoben und hinterher wieder geholt. Das erste Beispiel unten lief unter Unix fehlerfrei; das Bild war unverändert und auch das Word-Dokument war einwandfrei lesbar.

Unter Windows funktioniert dasselbe nur für Oracle-Versionen > 9.2, da vorher an jedes Zeilenende ein Zeilenvorschub (<CR><LF>) angehängt wurde.

Ab 10g werden Binärdateien im Byte-Modus geöffnet. Dies geschieht durch Angabe von 'RB', 'WB' und 'AB' statt 'R', 'W' und 'A' bei der FOPEN-Funktion.

Beispiele:

Auslesen und Schreiben eines Binärfiles

```

DECLARE
  v_file  UTL_FILE.FILE_TYPE;
  v_raw   RAW(32000);
BEGIN
  v_file := UTL_FILE.FOPEN('FOR_UTL_FILE', 'test.doc', 'rb', 24000);
  UTL_FILE.GET_RAW(v_file, v_raw, 32000);
  UTL_FILE.FCLOSE(v_file);
  v_file := UTL_FILE.FOPEN('FOR_UTL_FILE', 'testneu.doc', 'wb', 24000);
  UTL_FILE.PUT_RAW(v_file, v_raw);
  UTL_FILE.FCLOSE(v_file);
END;
```

Beispiel für den Einsatz von FGETATTR, FCOPY, FREMOVE, FRENAME, FGETPOS und FSEEK

```

DECLARE
  v_file      UTL_FILE.FILE_TYPE;
  v_exists    BOOLEAN;
  v_len       NUMBER;
  v_bs        NUMBER;
  v_buffer    VARCHAR2(2000);
  v_pos       PLS_INTEGER;
BEGIN
  -- Die Prozedur FGETATTR liest Länge, Blockgröße der Datei aus
  UTL_FILE.FGETATTR('FOR_UTL_FILE', 'test.txt', v_exists, v_len, v_bs);
  IF v_exists THEN
    DBMS_OUTPUT.PUT_LINE('Größe der Datei: ' || v_len || ' Bytes');
  ELSE
    DBMS_OUTPUT.PUT_LINE('Datei nicht gefunden');
    RETURN;
  END IF;

  -- die Datei test.txt wird kopiert
```

```
UTL_FILE.FCOPY('FOR_UTL_FILE', 'test.txt', 'FOR_UTL_FILE', 'test_2.txt');

-- und danach gelöscht
UTL_FILE.REMOVE('FOR_UTL_FILE', 'test.txt');

-- Die Kopie wird zum Lesen geöffnet
v_file := UTL_FILE.FOPEN('FOR_UTL_FILE', 'test_2.txt', 'r');

-- und die erste Zeile ausgelesen
UTL_FILE.GET_LINE(v_file, v_buffer);

-- Die Funktion FGETPOS ermittelt die Position des aktuellen Offsets
-- innerhalb des Files (also die Position des Zeilenendes)
v_pos := UTL_FILE.FGETPOS(v_file);
DBMS_OUTPUT.PUT_LINE('Position: ' || v_pos);

-- Mittels FSEEK wird die (relative) Position um 6 Zeichen zurück gesetzt
UTL_FILE.FSEEK(v_file, NULL, -(v_pos-6));

-- und der Rest der Zeile eingelesen
UTL_FILE.GET_LINE(v_file, v_buffer);
DBMS_OUTPUT.PUT_LINE(v_buffer);           -- Ausgabe "World!"

-- Hier wird eine absolute Position innerhalb des Files angegeben
UTL_FILE.FSEEK(v_file, v_len - 13);
UTL_FILE.GET_LINE(v_file, v_buffer);     -- Ausgabe "MuniQSoft"
DBMS_OUTPUT.PUT_LINE(v_buffer);

UTL_FILE.FCLOSE(v_file);
UTL_FILE.FRENAME('FOR_UTL_FILE', 'test_2.txt',
                'FOR_UTL_FILE', 'test_neu.txt');

END;
```

```
Ausgabe:
Größe der Datei: 53 Bytes
Position: 13
World!
MuniQSoft
```

Anmerkungen:

- Beim ersten Beispiel ist es wichtig, beim FOPEN eine maximale Zeilenlänge anzugeben, die groß genug ist für GET_RAW bzw. PUT_RAW, weil sonst ein UTL_FILE.READ_ERROR bzw. UTL_FILE.WRITE_ERROR ausgelöst wird.
- Ist eine Datei nicht vorhanden, so hat im zweiten Beispiel v_exists den Wert NULL, nicht FALSE.
- Bei FCOPY kann optional noch mit angegeben werden, von welcher Zeile (5. Parameter) bis zu welcher Zeile (6. Parameter) kopiert werden soll.
- Bei FRENAME kann optional als letzter Parameter noch angegeben werden, ob eine eventuell bereits vorhandene Datei dieses Namens überschrieben werden soll (TRUE) oder nicht (FALSE); der Default ist FALSE. Bei FALSE führt es zu einem Laufzeitfehler (UTL_FILE.RENAME_FAILED), wenn die Datei schon existiert.