Mail direkt aus Oracle versenden - komfortabel und mit Umlauten

Kunde:	DOAG*News*
Ort, Datum:	Artikel im Heft Q1 / 2005
Thema / Themen:	**Artikel von merlin.zwo**
Projekt:	**Mail direkt aus Oracle versenden – komfortabel und mit Umlauten**
Autor:	**Jochen Kutscheruk**

- Oracle & Technologien
- Systementwicklung
- Individuelle Lösungen

Einleitung
Schon seit Jahren kann man über das utl_smtp-Package Mails direkt aus einer Oracle-Datenbank versenden. Aber wirklich komfortabel ist dieses Package nicht. Um genau zu sein, handelt es sich nur um einen Wrapper um das utl_tcp-Package, damit nicht auf unterster Protokollebene mit einem Mailserver kommuniziert werden muß. Leider nimmt das Package keinerlei Rücksicht auf 8Bit-Zeichensätze, die außerhalb (US-)Amerikas verwendet werden.

Erst in Oracle 10*g* wird es durch das utl_mail-Package ersetzt bzw. ergänzt. Hiermit können tatsächlich komfortabel Mails versandt werden. Allerdings sind noch nicht alle Oracle-Datenbanken auf die neueste Version umgestellt, die Verwendung von utl_mail ist somit nicht immer möglich. Um auch in Nicht-10*g*-Datenbanken eine brauchbare Funktionalität zu erhalten, wird im Folgenden eine Lösung für einen komfortableren Mailversand vorgestellt.

Kommunikation mit dem Mailserver
Im Prinzip ist es recht einfach, über das utl_smtp-Package und einen Mailserver Mails zu versenden:

- Öffnen einer Verbindung zum Mailserver:
  c := utl_smtp.open_connection(*ip_of_mailserver*);
  -- c ist vom Typ utl_smtp.connection und wird für die weitere Kommunikation benötigt
- Begrüssen des Mailservers:
  utl_smtp.helo(c, 'absende_db@mydomain.de');
  -- Hier wird der Name des absendenden Datenbankservers bzw. der Instanz übergeben (frei wählbar)
- Die EMail-Adresse des Absenders bekanntgeben:
  utl_smtp.mail(c, 'ichbins@mydomain.de');
- Die Empfänger übergeben, einen nach dem anderen. An dieser Stelle wird noch nicht unterschieden, wer die Mail 'To:', 'Cc:' oder 'Bcc:' erhält! Dies erfolgt erst später.
  utl_smtp.rctp(c, 'jemand@yourdomain.de');
  utl_smtp.rcpt(c, 'nochjemand@yourdomain.de');
  utl_smtp.rcpt(c, 'anderer@yourdomain.de');

- Jetzt kommen die Daten:
  utl_smtp.open_data(c);
- **8Bit-Zeichensatz ermöglichen**Zuerst bekannt geben, daß jetzt 8Bit-Daten im ISO8859-15-Zeichensatz kommen (können). Ansonsten geht der Mailserver implizit von einem 7Bit-Zeichensatz aus.
  utl_smtp.write_data(c, 'MIME-version: 1.0' || utl_tcp.CRLF);
  utl_smtp.write_data(c, 'Content-Type: text/html; charset=ISO-8859-15' ||utl_tcp.CRLF);
  utl_smtp.write_data(c, 'Content-Transfer-Encoding: 8bit' ||utl_tcp.CRLF);
- Nun wird dem Mailserver bekanntgegeben, wer Absender (From:), Empfänger (To:), Cc: und Bcc: ist. Es müssen alle oben genannten Empfänger aufgeführt werden. Zusätzlich kann an dieser Stelle für jede Adresse ein Klartextname angegeben werden, der beim Empfänger im Mailprogramm angezeigt wird.
  utl_smtp.write_data(c, 'From: “Ich Bins“ <ichbins@mydomain.de>'||utl_tcp.CRLF);
  utl_smtp.write_data(c, 'To: jemand@yourdomain.de'||utl_tcp.CRLF);
  utl_smtp.write_data(c, 'Cc: nochjemand@yourdomain.de'||utl_tcp.CRLF);
  utl_smtp.write_data(c, 'Bcc: “Ein Anderer“ <anderer@yourdomain.de>'||utl_tcp.CRLF);
- Jetzt das Subject:
  utl_smtp.write_data(c, 'Subject: Das ist ein Betreff'||utl_tcp.CRLF);
  utl_smtp.write_data(c, utl_tcp.CRLF); -- Es werden zwei CR/LF gefordert
- Schließlich der eigentliche Mailbody, welcher als raw_data übergeben wird. Hier können jetzt problemlos Sonderzeichen geschickt werden, da wir zu Beginn den gewünschten Zeichensatz angegeben hatten!
  utl_smtp.write_raw_data(c, 'Nachricht mit Sonderzeichen: ÖÄÜöäüß€!');
- Und zum Schluß die Verbindung wieder ordentlich beenden:
  utl_smtp.close_data(c);
  utl_smtp.quit(c);

Sonderzeichen auch im Header und Subject

So weit, so gut. Nun kommen zwar im Mailbody alle Sonderzeichen korrekt an, nicht jedoch im Subject oder bei den Mailadressen. Der Adressat “Günther Müller“ <guenther.mueller@yourdomain.de> würde unleserlich beim Empfänger ankommen, da die Umlaute hier immer noch nicht korrekt behandelt werden.
Leider hilft es nicht, diese Daten ebenfalls einfach per utl_smtp.write_raw_data zu senden. Daß im Betreff oder bei den Adressen ebenfalls Umlaute vorkommen können, muß man dem Mailserver separat mitteilen. Das Prinzip soll am Beispiel des Subjects verdeutlicht werden:

((Beginn Programmcode))
```
utl_smtp.write_data(c, 'Subject: =?ISO-8859-
15?Q?'||utl_raw.cast_to_varchar2
(utl_encode.quoted_printable_encode(utl_raw.cast_to_raw('Dös
is dös Subject'))))||'?='||utl_tcp.crlf);
```
((Ende Programmcode))

Auf den ersten Blick sieht dies kompliziert aus, ist es aber nicht. Zuerst wird der Subject-Text nach Raw gewandelt (utl_raw.cast_to_raw), damit der String über utl_encode in ein „quoted printable“ gewandelt werden kann. Das kann durch den Mailserver korrekt verarbeitet werden. Anschliessend werden diese Raw-Daten wieder nach Varchar2 umgewandelt (utl_raw.cast_to_varchar2), damit utl_smtp.write_data aufgerufen werden kann (hier wird ein Varchar2 als Parameter erwartet). Schliesslich wird dem Mailserver noch ein Tipp mitgegeben, an welcher Stelle sich der umgewandelte String befindet (zwischen „=?“ und „?=“) und um welchen Zeichensatz es sich handelt (ISO-8859-15, damit auch Eurozeichen mitgeschickt werden können). Dieser String wird dann an den Mailserver übergeben, der die Umlaute nun korrekt weiterleitet.

Für die Mailadressen gibt es zwei Möglichkeiten der Adressierung: guenther.mueller@yourdomain.de (einfache Adressierung) oder aber “Günther Müller“ <guenther.mueller@yourdomain.de> (Adressierung mit zusätzlichem Klartextnamen)

Bei der ersten Variante muß nichts unternommen werden, da nur 7Bit-Zeichen in der Mailadresse vorkommen können. Bei der zweiten Variante würde das oben genannte Verfahren prinzipiell auch funktionieren, jedoch haben einige Mailprogramme Probleme damit, wenn der „quoted printable“-Text die eigentliche Mailadresse umfasst (also bis zum schliessenden „>“ geht). Folglich darf man die Umwandlung nur bis zum einleitenden „<“ durchführen. Das Beispiel mit „Günther Müller“ muß folgendermassen an den Mailserver gesendet werden:

((Beginn Programmcode))
```
utl_smtp.write_data(c, 'To: =?ISO-8859-
15?Q?'||utl_raw.cast_to_varchar2(utl_encode.quoted_printable_e
ncode(utl_raw.cast_to_raw(“Günther
Müller“)))||'?='||'<guenther.mueller@yourdomain.de>'||utl_tcp.
crlf);
```
((Ende Programmcode))

Lediglich „Günther Müller“ wird in „quoted printable“ umgewandelt, die eigentliche Mailadresse <guenther.mueller@yourdomain.de> bleibt unangetastet.

Das fertige Package

Als letzter Schritt wird das Verfahren in ein Package gekapselt. Damit entsteht eine wiederverwendbare Komponente, die den Aufwand für den Versand von Mails aus der Datenbank erheblich reduziert.

Das entstandene Package wurde gleich so erweitert, daß sowohl ein einzelner Empfänger als auch eine PL/SQL-Table mit einer Empfängerliste übergeben werden kann.
Nachfolgend ist der Quellcode des Packages als Zusammenfassung dargestellt.

((Beginn Programmcode))

```
create or replace PACKAGE M2MAIL as
  mailserver varchar2(64) := '192.168.60.4';
  absende_db varchar2(64) := 'm2test.m2db.merlin-zwo.de';

  TYPE empfaenger_rec IS RECORD
  (empfaenger varchar2(64),
   cc_bcc varchar2(3)
  );
  TYPE empfaenger_tab IS TABLE OF empfaenger_rec
   INDEX BY BINARY_INTEGER;

  -- "senden" kann entweder mit einer Empfängerliste (TYPE empfaenger_tab)
  -- aufgerufen werden oder mit einem einzelnen Empfänger. Die Absender-
und
  -- Empfängersyntax muß natürlich korrekt sein, es wird hier keine weitere
  -- Prüfung vorgenommen.
  --
  -- Bei Verwendung einer Empfängerliste muß in der Spalte cc_bcc folgendes
  -- übergeben werden:
  -- - NULL oder 'To' : Das ist der eigentliche Emfänger
  -- - 'Cc' : An diesen Empfänger geht die Mail cc
  -- - 'Bcc' : An diesen Empfänger geht die Mail bcc
  --
  -- Der Absender/Empfänger kann auch mit ausführlichem Namen angegeben
werden,
  -- also z.B. '"Jochen Kutscheruk" <jochen.kutscheruk@merlin-zwo.de>'
  -- Dadurch erscheint dann im Mailprogramm des Empfängers der
Klartextname.
  --
  -- Wenn die Mail erfolgreich versandt wurde wird TRUE zurückgegeben und
der
  -- Fehlertext ist NULL. Ansonsten wird FALSE zurückgegeben und die
Variable
  -- „fehlertext“ enthält den SMTP-Fehler im Klartext.
  --
  -- Bevor Fragen kommen: Attachments sind mit dieser Package NOCH NICHT
möglich!

function senden(absender in varchar2, empfaenger in out empfaenger_tab,
                  betreff in varchar2, mailbody in varchar2,
                  fehlertext out varchar2)
   return boolean;
```

```
  function senden(absender in varchar2, empfaenger in varchar2,
                  betreff in varchar2, mailbody in varchar2,
                  fehlertext out varchar2)
   return boolean;
end m2mail;
/

create or replace PACKAGE BODY M2MAIL as
  c utl_smtp.connection;

  PROCEDURE send_subject(subject IN VARCHAR2) IS
    -- Übergabe des Subjects an den Mailserver

  BEGIN
    utl_smtp.write_data(c, 'Subject: =?ISO-8859-15?Q?'||
     utl_raw.cast_to_varchar2(utl_encode.quoted_printable_encode
      (utl_raw.cast_to_raw(subject)))||'?='||utl_tcp.crlf);
  END send_subject;

  PROCEDURE send_adresse(name IN VARCHAR2, adresse IN VARCHAR2) IS
    -- Übergabe einer Mailadresse an den Mailserver
    -- Die Übergabevariable „name“ kann die Werte „From:“, „To:“,
    -- „Cc:“ oder „Bcc:“ annehmen.
    klammerpos number;
  BEGIN
    -- Absender / Empfänger nur bis zur ersten '<' umkodieren,
    -- danach muß sowieso alles US7ASCII sein!
    klammerpos := instr(adresse,'<');
    if klammerpos > 1 then  -- Die Klammer steht nicht am Anfang
      utl_smtp.write_data(c, name||': =?ISO-8859-15?Q?'||
       utl_raw.cast_to_varchar2(utl_encode.quoted_printable_encode
        (utl_raw.cast_to_raw(substr(adresse,1,klammerpos-1))))
       ||'?='||substr(adresse,klammerpos)||utl_tcp.crlf);
    else
      utl_smtp.write_data(c, name||': '||adresse||utl_tcp.CRLF);
    end if;
  END send_adresse;
```

```
  function mailadresse(adresse in varchar2) return varchar2 is
    -- Die reine Mailadresse extrahieren (z.B wenn als Empfänger
    -- '"Jochen Kutscheruk" <jochen.kutscheruk@merlin-zwo.de>' angegeben
ist,
    -- dann wird 'jochen.kutscheruk@merlin-zwo.de' zurückgegeben. Diese
Adresse
    -- wird für den Header (utl_smtp.rcpt und utl_smtp.mail) benötigt.
    klammerpos1 number;
    klammerpos2 number;
  begin
    klammerpos1 := instr(adresse,'<');
    klammerpos2 := instr(adresse,'>');
    if klammerpos1 != 0 and klammerpos2 != 0 then
      return(substr(adresse,klammerpos1+1,klammerpos2-klammerpos1-1));
    else
      return(adresse);
    end if;
  end;

  function senden(absender in varchar2,empfaenger in out empfaenger_tab,
                  betreff in varchar2,mailbody in varchar2,
                  fehlertext out varchar2)
   return boolean is
  -- Senden mit Empfängerliste (empfaenger_tab)
  BEGIN
    if empfaenger.COUNT = 0 then
      fehlertext := 'Leere Empfängerliste übergeben!';
      return(FALSE);
    end if;

    c := utl_smtp.open_connection(mailserver);
    -- Verbindung herstellen
    utl_smtp.helo(c, absende_db);
    -- Absender bekanntgeben
    utl_smtp.mail(c, mailadresse(absender));
    -- Alle Empfänger bekanntgeben
    for i in empfaenger.FIRST..empfaenger.LAST loop
      if empfaenger.EXISTS(i) then  -- Könnte evtl. leer sein!
        if empfaenger(i).empfaenger is not null then
          utl_smtp.rcpt(c, mailadresse(empfaenger(i).empfaenger));
        end if;
      end if;
    end loop;
    -- Jetzt der eigentlich Datenteil
    utl_smtp.open_data(c);
    -- 8Bit einstellen
    utl_smtp.write_data(c, 'MIME-version: 1.0' || utl_tcp.CRLF);
    utl_smtp.write_data(c, 'Content-Type: text/html;charset=ISO-8859-15'||
                           utl_tcp.CRLF);
    utl_smtp.write_data(c, 'Content-Transfer-Encoding: 8bit' ||
                           utl_tcp.CRLF);
    -- Nochmals den Absender angeben
    send_adresse('From', absender);
```

```
    -- Jetzt nochmal alle Empfänger, diesmal aber mit der Angabe,
    -- ob „To:“, „Cc:“ oder „Bcc:“ gemailt wird
    for i in empfaenger.FIRST..empfaenger.LAST loop
      if empfaenger.EXISTS(i) then
        if empfaenger(i).empfaenger is not null then
          if initcap(empfaenger(i).cc_bcc)
             not in ('To','Cc','Bcc') then
            empfaenger(i).cc_bcc := 'To'; -- Default auf „To:“
          end if;
          send_adresse(empfaenger(i).cc_bcc,
                       empfaenger(i).empfaenger);
        end if;
      end if;
    end loop;
    -- Subject senden
    send_subject(betreff);
    utl_smtp.write_data(c,utl_tcp.CRLF);
    -- Jetzt kommt erst der eigentliche Mailbody
    utl_smtp.write_raw_data(c, utl_raw.cast_to_raw(mailbody));
    -- Und Verbindung wieder ordentlich beenden
    utl_smtp.close_data(c);
    utl_smtp.quit(c);
    fehlertext := NULL;
    return(TRUE);  -- Wenn bis hierher keine Exception kam...
  EXCEPTION
    WHEN utl_smtp.transient_error OR
         utl_smtp.permanent_error THEN
      -- Es ist ein Fehler aufgetreten, Verbindung schliessen
      BEGIN
        utl_smtp.quit(c);
      EXCEPTION
        WHEN utl_smtp.transient_error OR
             utl_smtp.permanent_error THEN
          -- Wenn beim utl_smtp.quit eine Exception kommt, dann diese
ignorieren
          NULL;
      END;
      fehlertext := 'Fehler beim senden der EMail, Fehlercode: '||
                    sqlerrm;
      return(FALSE);
  END senden;
```

```
  function senden(absender in varchar2, empfaenger in varchar2,
                  betreff in varchar2, mailbody in varchar2,
                  fehlertext out varchar2)
   return boolean is
    empftab empfaenger_tab;
  -- Funktion „senden“ überlagert für Versand nur an einen einzelnen
Empfänger.
  begin
    empftab(1).empfaenger := empfaenger;
    empftab(1).cc_bcc := 'To';
    -- Die Funktion „senden“ wird einfach mit einer PL/SQL-Table
    -- mit nur einer Zeile aufgerufen
    return(senden(absender,empftab,betreff,mailbody,fehlertext));
  end senden;
END m2mail;
```

((Ende Programmcode))

Das Package kann von unserem Webserver unter
http://www.merlin-zwo.de/downloads heruntergeladen werden.

Mit diesem Package können noch keine Attachments verschickt werden. Prinzipiell wäre dies aber durch eine entsprechende Erweiterung des Package möglich. Es müßte nur im Mailbody ein neuer Abschnitt mit dem korrekten MIME-Type erzeugt werden, in den das Attachment (korrekt codiert) übergeben würde. Zu diesem Thema sind auch im Metalink einige Hinweise zu finden.

Sollten Fragen zu diesem Artikel bestehen - ich stehe Ihnen gerne zur Verfügung:

Ihr Ansprechpartner:	**Kontakt**
Jochen Kutscheruk	jochen.kutscheruk@merlin-zwo.de
Geschäftsführer, Systementwickler	Tel.: 0721 / 790 71 71 Fax: 0721 / 790 71 99
merlin.zwo InfoDesign GmbH & Co. KG Taglöhnergärten 43 76228 Karlsruhe	www.merlin-zwo.de
merlin.zwo InfoDesign GmbH & Co. KG Karmelstraße 9 75378 Bad Liebenzell	