

Der Entwickler 05/2000

Native und Free - NCOCI8 - native Komponenten zum Zugriff auf Oracle 8

von Henry Wolf

Seit geraumer Zeit erfreuen sich native Zugriffskomponenten für Delphi zum Zugriff auf Oracle großer Beliebtheit. Insbesondere das Produkt DOA der holländischen Firma Allround Automations VOF (<http://www.allroundautomations.nl>) hat in der Vergangenheit eine große Verbreitung erfahren. Ein neues Produkt welches dem Geist der Zeit - Open Source - Rechnung trägt ist NCOCI8, eine Freeware (mit Source) - Lösung des russischen Programmierers Dmitry Arefiev.

NCOCI8 stellt eine objektorientierte Kapselung der Oracle8 / 8i - OCI Schnittstelle in der Sprache Delphi dar und unterstützt die Delphi - Versionen 3, 4 und 5. Die OCI-Schnittstelle stellt die von ORACLE zur Verfügung gestellte, plattformunabhängige und veröffentlichte API zur Implementierung von Anwendungen auf der Basis des RDBMS Oracle dar. Sowohl die BDE als auch ODBC und die meisten Implementierungen von nativen Zugriffen auf Oracle basieren auf dieser API. Unter Windows wird diese Schnittstelle in Form von DLL's (Ora73.DLL - Oracle73 -, OCI.DLL - Oracle8/8i -) implementiert. Im Ergebnis der Kapselung werden in diesem Package grundsätzlich 5 Komponenten zur Realisierung des Datenzugriffes zur Verfügung gestellt:

- **OCIDatabase**: Kapselung der Datenbankverbindung
- **OCITransactionManager**: Komponente zur Konfiguration der Transactionen / Transaction-Management
- **OCIQuery**: Komponente zum Zugriff via SQL
- **OCIStoredProc**: Komponente zum Zugriff auf Stored Procedures
- **OCIUpdateSQL**: Komponente zur Implementierung von Cached Updates

1. Datenzugriffskomponenten

Die Datenzugriffskomponenten sind vollständig *TDataSet*-kompatibel implementiert und können so problemlos mit den datensensitive Komponenten von Delphi verwendet werden.

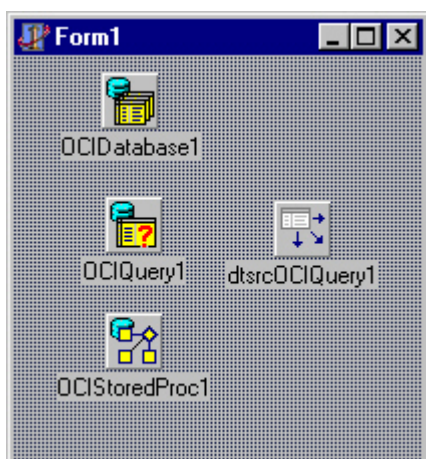


Abb.1 Datenzugriffskomponenten

Bei der Implementierung von *TOCIQuery* ist insbesondere die Eigenschaft *RecordCountMode* zu beachten. Hierbei wird konfiguriert, wie das Ergebnis von *TOCIQuery.RecordCount* zu interpretieren ist.

- **cmFetched**: Anzahl der aktuell auf den Client übertragenen Datensätze
- **cmExactAlways**: exakte Anzahl zu jedem Zeitpunkt
- **cmExactonOpen**: exakte Anzahl zum Zeitpunkt des öffnens der *TQuery*

Neben den bereits von der Komponente *TQuery* bekannten Ereignissen sind in der Komponente *TOCIQuery* weitere hochinteressante Ereignisse zu deren Behandlung implementiert so z.B.

- BeforePrepare
- AfterPrepare
- BeforeUnPrepare
- AfterUnPrepare
- BeforeRecordCount
- AfterRecordCount

Gleiches gilt für die Komponente *TStoredProc* und besonders für die Komponente *TDatabase* wo zahlreiche Ereignisroutinen zur Verfügung gestellt werden.

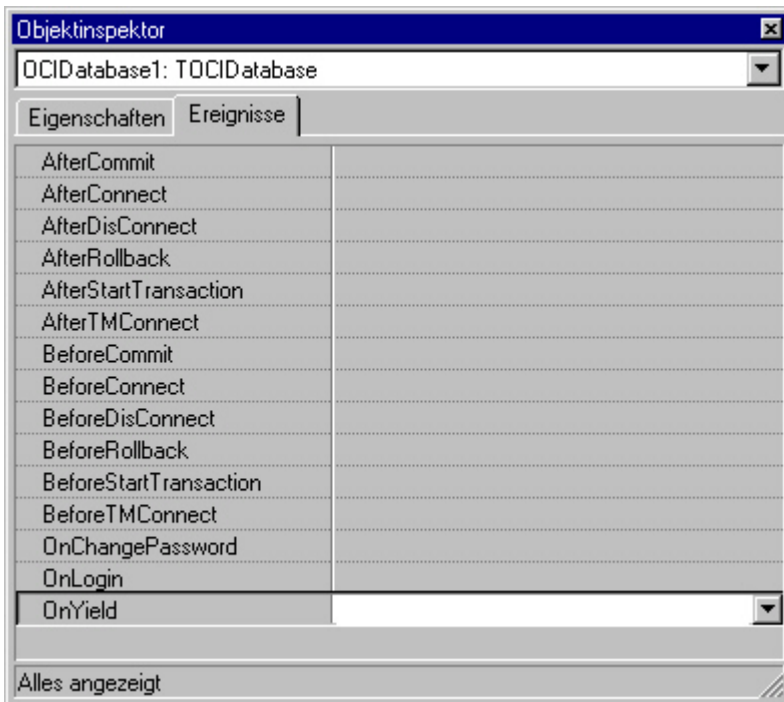


Abb.2 Ereignisroutinen der Komponente TDatabase

Unbequem ist bei der Nutzung der BDE der Zugriff auf Stored Procedures, welche in Oracle-PL/SQL-Packages implementiert sind. Man muß manuell ermitteln wie Package bzw. Procedure heißt und diese in das Feld *StoredProcName* mit Hand eintragen. Die Komponente *TOCStoredProc* erhöht hierbei wesentlich den Komfort des Zugriffes auf solche Procedures. Man wählt via der Eigenschaft *OPackagename* ein Package aus und unter *OProcedureName* werden einem die Procedures / Funktionen in dem Package angeboten.

Nicht implementiert sind derzeitig extra Komponenten für Nested Tables und Objekte. Hierbei sei auf die TDataset-Kompatibilität verwiesen. Außerdem werden auch heute noch die meisten Datenbanken nach der 3 Normalform des relationalen Datenbankmodells implementiert.

2. Transaktionen

Die Implementierung der Komponenten öffnet den Delphi-Entwicklern eine große Menge von Möglichkeiten, um Funktionen der Datenbank Oracle optimal auszunutzen. Als Beispiel hier kurz eine Anmerkung zum Transaktionsmanagement. Dieses stellt bei der Entwicklung von Oracle-Anwendungen auf Basis der BDE insofern ein großes Problem dar, als das die Zuweisung von speziellen Rollback-Segmenten für Transaktionen innerhalb eines Delphi-Programmes via SQL / *TQuery* nicht möglich ist.

Nehmen wir an, wir wollen unserer aktuellen Transaktion das Rollback Segment RB_BIGTRANS zuweisen, müßte die Syntax in SQL*Plus müßte wie folgt aussehen und:

```
SQL> COMMIT; /* Ende der letzten, Begin der neuen Transaction
SQL> SET TRANSACTION USE ROLLBACK SEGMENT RB_BIGTRANS;
SQL> INSERT ..... UPDATE ..... DELETE
SQL> COMMIT; /* Ende der letzten, Begin einer neuen Transaction
```

Die Zuweisung des Rollback Segmentes via SET Transaction wird nicht als SQL-Statement durch die Query akzeptiert. Bleibt noch *DBIExecDirect*, mittels welchen SQL-Statements an eine SQL-Server übermittelt

werden können ohne von der BDE interpretiert zu werden. Dies akzeptiert diese SQL-Statements natürlich erzielen aber nicht in jedem Falle die gewünschte Wirkung. Erfolgreich lässt sich diese Art der Übermittlung von SQL-Befehlen bei der Bearbeitung von Metadaten und z.B. bei der Erteilung von Rechten einsetzen.

```
procedure TFormExecuteRightManagement.PrivProcExecGrant;
begin
  Check ( DBIQExecDirect ( Database1.handle , QRYLANGSQL ,
    PChar( 'GRANT EXECUTE ON '+ NAME + 'TO' + Nutzer ), nil ));
end;
```

In NCOCI8 stellt dies kein Problem dar. Über die Komponente *OCITransactionManager* kann eine Transaktion vollständig konfiguriert werden und natürlich auch ein Rollbacksegment explizit zugeordnet werden. Insbesondere bei vereinzelt großen Transaktionen erleichtert diese Möglichkeit der einfachen Konfiguration die Einsatzmöglichkeiten von Delphi als Werkzeug.



Abb.3 die Komponente OCITransactionManager

Die Komponente *OCIDatabase*, welche weitestgehend von den Aufgaben her der VCL-Komponente *TDatabase* und *TSession* entspricht, wird ein solche konfigurierte Komponente *OCI-TransactionManager* über die Eigenschaft *TransactionManager* zugewiesen und somit die Steuerung von "konfigurierten" Transaktionen innerhalb der Datenbankverbindung ermöglicht. Ist das Transactionsumfeld eingestellt bzw. konfiguriert können via Code Transaktionen gestartet und abgeschlossen werden.



Abb.4 Zuweisung eines OCITransactionManager zu einer OCIDatabase

```

procedure TNC08ConMainFrm.ProcStartTransaktion;
begin
  if ( database.InTransaction ) then
    database.Commit
  else
    database.StartTransaction;
end;

```

```

procedure TNC08ConMainFrm.ProcCommit;
begin
  database.Commit;
end;

```

```

procedure TNC08ConMainFrm.ProcRollback;
begin
  database.Rollback;
end

```

3. Exceptionhandling

Zur Behandlung von Fehlern, welche durch Datenzugriffe u.s.w. auf der Basis von NCOC18 verursacht werden, sind auf Basis der Klasse *EOCIErrorBase* 3 Exceptionklassen implementiert:

- **EOCIDBError**: Allgemein in NCOC18 auftretende Warnings / fehler

- **EOCINativeError**: ORACLE-Server-Fehler
- **EOCISystemError**: Allgemeine Fehler, BUG's in NCOCI

Hierbei ist insbesondere *EOCINativeError* interessant, da wir via der verfügbaren Eigenschaften an die originale Fehlernummer bzw. Fehlermeldung des ORACLE-Servers herankommen und diesen ungefiltert behandeln können.

E.Errors[i].ErrorCode Typ sb4 (= integer)- numerischer Fehlercode -
 E.Errors[i].Message Typ String - ORACLE Server - Fehlermeldung -
 E.Errors[i].Level Typ Longint - Fehlerlevel / Fehlerschwere -

Eine auf dem Exceptionhandling basierende Transaktionsbehandlung müsste in unserem Falle aussehen wie in Listing 2.

```
try
  if ( database.InTransaction ) then
    database.Commit
else
  database.StartTransaction;
  database.commit;
except
  on E: EOCINativeError
  do begin
    for i := 0 to E.ErrorCount -1 do
      MessageDlg( E.Errors[i].Message , mtError, [mbOK], 0);
      database.rollback;
    end;
  end;
end;
```

4. INTERNAL-Login

Eine weitere interessante Möglichkeit, welche durch NCOCI8 eröffnet wird, ist die Nutzung sogenannter INTERNAL-Logins zur Implementierung von Lösungen zur Administration von Datenbanken, wo ein Teil der Aufgaben (Anlegen, Starten, Beenden der Datenbank u.a.) einen solchen benötigt. Ein solcher INTERNAL - Login wird auf der Basis sogenannter Administrationsuser erstellt.

Im ersten Schritt legt man ganz normal z.B. einen Nutzer OEMMAN an und weist ihm die Rollen CONNECT, RESOURCE und DBA zu. Im nächsten Schritt geht man in die INIT.ORA der Instance und bearbeitet den Parameter wie folgt:

```
remote_login_password = shared
wird geändert in
remote_login_password = exclusive
```

Dadurch wird es ermöglicht, außer dem INTERNAL Nutzer anzulegen, welche Remote sogenannte internal-Zugriffe auf die Datenbank realisieren können. Zur Aktivierung dieses Parameters ist die Datenbank neu zu starten. Nach dem Neustart wird in der Kommandozeile das Tool svrmgrl (Oracle8i, svrmgr30 ORACLE 8.0.x) und unserem Nutzer die INTERNAL-adäquaten Rechte zugewiesen.

```
svrmgrl> connect internal as sysdba;
svrmgrl> grant sysdba to oemman;
svrmgrl> grant sysoper to oemman;
```

Der Implementierung von Administrationsaufgaben bis hin zum Performancetuning (auch Remote) steht nun nichts mehr im Wege.

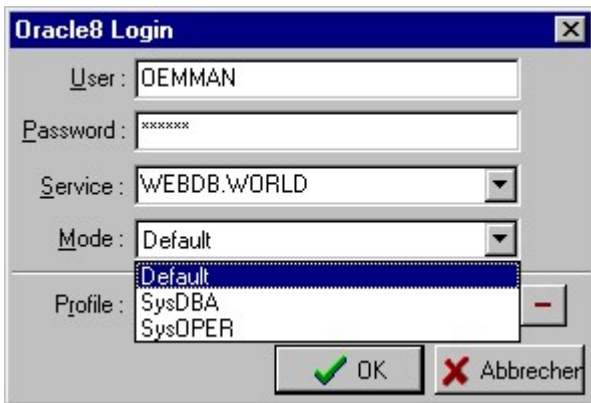


Abb.4 Standard-Anmeldungsdialog einer TOCIDatabase-Komponente

5. Initialisierung der OCI-Schnittstelle

Bei den meisten Tools ist es notwendig bzw. sinnvoll die Implementierung der Datenbankschnittstelle beim Start des Programmes statisch zu initialisieren. In NCOC18 besteht die Möglichkeit, die Implementierung der OCI-Schnittstelle dynamisch zu laden bzw. aus dem Speicher zu entfernen. Hierfür wurden die im Folgenden aufgeführten Prozeduren implementiert.

```
procedure InitOCI;  
procedure TerminateOCI;
```

6. Fazit

Im Laufe der Zeit ist mit dem Produkt NCOC18 ein gutes Werkzeug zum nativen Zugriff von Delphi auf Oracle8 entstanden. Erfahrene Delphi / Oracle-Entwickler können dieses Tool problemlos erfolgversprechend in Projekten einsetzen und an der Weiterentwicklung, sei es durch Hinweise oder Testberichte, mitwirken. Alle anderen Entwickler sollten das Produkt beobachten. Ist die Hilfe vervollständigt und der eine oder andere kleinere Bug, welcher nicht immer von NCOC18 sondern auch häufig von der OCI-API versucht wird, beseitigt steht allen Entwicklern eine sehr gute und kostengünstige Lösung zur Verfügung deren Weiterentwicklung durch eigenes Wirken nur besser werden kann. Auf der Leser-CD zum Heft befindet sich die Version 0.7.5 vom 04.06.2000.