# 7 Bibliotheken

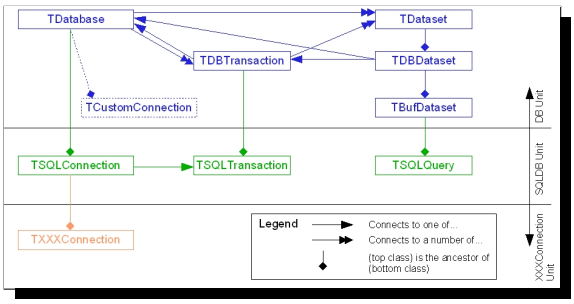
## 7.1 SQLdb

### 7.1.1 Beschreibung

#### Einleitung

SQLdb wird speziell für Serverbasierende Datenbanken verwendet und besteht im Wesentlichen aus Komponenten für die Verbindung (TxxxConnection), für das Verwalten von Transaktionen (TSQLTransaction) und dem Verwalten von Datenmengen (TSQLQuery). Für Clientdatenbanken (Dbase, FoxPro,...) sind die Komponenten unter 'Client Access'[1] vorgesehen.

Besonders die TSQLQuery ist eine mächtige Komponente, die einige automatismen eingebaut hat, die zwar das Leben erleichtern sollen, aber oft das Gegenteil bewirken können. In diesem Kapitel wollen wir uns die Komponenten einmal genauer ansehen. Das Bild aus der Wiki aus dem englischen Lazarusforum[LazEn][2] erklärt schön, wie die Komponenten zusammenhängen und in welchen Units sie sich befinden.




#### Debbuging von SQLdb

SQLdb ist ein Teil der FCL und nicht direkt von Lazarus. Die FCL wird standardmässig nicht mit den für den Debugger notwendigen Informationen kompiliert, da der Kode so kompakter und kleiner ist. Wenn man für die Fehlersuche es anders benötigt, so muß man die fcl-db neu kompileren. Dazu muß man die kompletten FCL Sourcen haben, dann kann man in das Verzeichnis 'fpc/packages/fcl-db' gehen und mit 'make clean all OPT='-gl'' das Paket neu kompiliren, anschliessen die neuen PPU's über die alten kopieren. Die Fehlersuche wird aber nur Personen empfohlen, die entsprechendes Wissen über die SQLdb Komponenten haben.

#### Active, Open oder ExecSQL

... das ist hier die Frage ? Um diese Frage zu beantworten, muß man sich vor Augen halten, was man von der Komponente will. Mittels dem Befehl Open fordert man eine Datenmenge, die

[1] Siehe Kapitel 7.2.1 auf Seite 84
[2] *http://wiki.lazarus.freepascal.org/SQLdb_Programming_Reference*

Kardinalität[3] einer Datenmenge[4] kann auch Null sein, an. Mit ExecSQL wird nur eine Aktion angefordert ohne das eine Datenmenge zurück erwartet wird. Somit ist klar, das bei allen Daten liefernden Statements das Open zu verwenden ist. Welche Statements in SQL liefern überhaupt Datenmengen zurück ? Eigentlich gilt das nur für das 'SELECT' Statement, alle anderen ('INSERT', 'UPDATE', 'DELETE', ...) führen etwas aus, liefern aber keine Daten zurück. Ob man jetzt 'Open' verwendet oder 'Active:=true;' macht ist letztlich egal, 'Open' führt genau dieses Statement aus.

Eine Besonderheit ist die Behandlung von 'Stored Procedure' und 'Functions' auf SQL-Servern. Diese können eine Kombination im Verhalten darstellen. Dort ist dann ein ExecSQL angebracht und es können Datenmengen zurückgeliefert werden.

Zusammenfassung:

- Open, Active: Bei der Verwendung von 'SELECT'
- ExecSQL: Für alle anderen Statements

### Wie kommen die geänderten Daten in die Datenbank

Eine änderung der Datenmenge alleine ist nicht ausreichend, um diese Änderung auch in der Datenbank sichtbar zu machen. Prinzipiell muß man jetzt zwei Wege unterscheiden. Einerseits kann man Änderungen im Zuge einer Transaktion in der Datenbank festschreiben durch das Abschliessen der Transaktion oder auch durch das dezitierte schreiben durch ApplyUpdates. Ich bin der Meinung, das man sich für einen der Wege entscheiden sollte, wenn man eine Datenmenge öffnet beziehungsweise anfordert. Arbeitet man mit Transdaktionen, dann soltte man ohne zwingenden Grund nicht mit ApplyUpdates das Transaktionsmanagment stören. Anderseits wenn man ohne explzite Transaktionen arbeitet, also Datenmengen öffnet ohne vorher Transaktionen geöffnet zu haben, dann darf man nicht vergessen die Änderungen entweder mittels ApplyUpdates zu übernehmen oder mittels CancelUpdates zu verwerfen. Vergisst man auf ApplyUpdates so ist dieses schlimmer als auf CancelUpdates zu vergessen. Denn ohne dem ApplyUpdates sind die Daten bei den meisten Datenbanken ganz einfach nicht in die Datenbank eingearbeitet und somit verloren.

### Filtern, aber wo ?

Die Standardantwort ist im Stile von Radio Eriwan: 'Dort wo es sinnvoll ist'. Dazu muß man sich vor Augen halten, wo man eine Datenmenge überhaupt filtern kann. Dazu muß man unterscheiden in Desktop Datenbanken und Server Datenbanken. Bei Desktop Datenbanken kann die Frage schon obsolet sein, weil die Datenmenge sowieso nur lokal gefiltert werden kann. Bei Datenbankenservern schaut die Sachlage ganz anders aus. Denn die können Datenmengen sehr wohl, effizient vor verarbeiten und nur die wenigen Ergebnisse zurück transportieren. Damit wird am lokalen Rechner Netzwerkleistung, Speicher und Resourcen geschont. Somit kann

[3] siehe Kapitel 3.1.1 auf Seite 30
[4] siehe Kapitel 3.1.1 auf Seite 28

man hier dem filtern am Server den Vorzug geben. Werden aber Daten erst am lokalen Rechner verknüpft, so kann man oft nur lokal filtern. Somit ist klar, das es stark auf das Design der Applikation an kommt, was sinnvoll ist.

### Anzahl der Datensätze abfragen

Hier kann man generell zwei verschiedene Fälle unterscheiden. Einmal den Fall, das man wissen will, ob überhaupt Datensätze vorhanden sind und dem Fall, das man die Anzahl wissen will.

Generell ist es nur dann sinnvoll die Anzahl der Datensätze zu bestimmen, wenn eine Abfrage aktiv ist.

Ob Datensätze überhaupt vorhanden sind, kann man über die Abfrage von EOF[5] und BOF[6] machen. Sind beide vorhanden ('true') so muß die Datenmenge leer sein. Genau diese macht die Methode 'IsEmpty'. Somit kann man diese genau für diesen Fall verwenden.

Die Anzahl selbst der Datensätze, kann man theoretisch mittels der Eigenschaft 'RecordCount' abfragen. Alledings muß dazu auch der Datenbanktreiber[7] das unterstützen. Bis jetzt ist die Unterstützung auch nicht wirklich vorhanden. Weiters handelt es sich hier eher um eine Eigenschaft von Desktopdatenbanken, denn dort kann die Anzahl der Datensätze nicht anders festgestellt werden.

Die andere Variante die sich daher anbietet ist die SQL Abfrage selbst. So kann man mittels dem SQL-Statement `select count(row1) as Anzahl from table1 where ...` die Anzahl ermitteln.

### Navigieren durch eine Datenmenge

Für das Navigieren durch die Datenmenge stehen ein paar Befehle zur Verfügung. Mit 'first' kommt man zum ersten, mit 'last' zum letzten, mit 'next' springt man auf den nächsten und mit mit 'prior' zum vorhergehenden Datensatz. Größere Bewegungen kann man mit 'MoveBy' machen. Das funktioniert vorwärts mit positiven Zahlen, rückwärts mit negativen Zahlen. Allerdings muß man bedenken, das ein 'MoveBy' nicht zwingend die volle Distanz verfahren kann, wenn die Grenzen der Datenmenge erreicht werden. Wenn also ein EOF oder BOF nach dem 'MoveBy' ansteht, so wird nicht die volle Distanz erreicht worden sein, man weiß aber nicht um wie viel verfahren wurde.

### Was ist BOF und EOF

'BOF' bedeutet das man in der Datenmenge am Anfang, bei 'EOF' am Ende der Datenmenge steht. Wenn zum gleichen Zeitpunkt beide vorhanden sind, so ist das ein Zeichen, das die Datenmenge null ist.

[5] End of File - Anfang der Daten
[6] Beginn of File - Ende der Daten
[7] Ist genaugenommen die Verbindungskomponente