Norbert Kessel · SQLBase - Eine Einführung
www.forstbuch.de

alte ISBN 3-00-006604-7
neue ISBN 3-935638-05-1

Druck: www.business-copy.com

Vorwort

Dieses Buch will in zwei Gebiete einführen, zum einen in die Begriffswelt der relationalen Datenbank **SQLBase** und zum anderen in die Denkweise der Sprache **SQL** (Structured Query Language). Dabei werden zunächst die Eigenschaften der Datenbank sowie der damit verwendbaren Software (z. B. SQLTalk) beschrieben und danach die Bestandteile der Sprache SQL vorgestellt.

Dass es dabei durchaus zu Überschneidungen kommt, soll gleich hier vorweggenommen werden, denn einige wichtige Konzepte der SQLBase (z. B. UNLOAD/LOAD oder das Wissen um „pages“ für die Datenspeicherung in SQLBase) sind so elementar, dass sie am Anfang schon erläutert werden müssen. Allerdings versteht man beispielsweise die Anweisung UNLOAD erst dann richtig, wenn man mit einem Programm zur Betreuung der Datenbank (wie SQLTalk oder SQLConsole schon vertraut ist; um damit vertraut zu werden muss man aber üben und hierzu braucht man eine Datenbank und darin enthaltene Tabellen mit Daten. Aus diesem Grund wird sich eine Anweisung (hier zum Erstellen einer Tabelle) wie z. B.

```
create table ...
```

mehrfach in dem Buch finden, allerdings immer in der Form, dass sich die Beispiele (also alles, was in der vorstehenden Schrift formatiert ist) sofort nachvollziehen lassen. Hinzu kommt, dass alle SQL-Anweisungen in diesem Buches von der Homepage des Autors **http://www.forstbuch.de** kostenlos heruntergeladen werden könne (s. dort unter Download, Dateiname: SQLcode.ZIP).

Gliederung des Buches

Das Buch ist in die folgenden Teile gegliedert:

Kapitel 1: führt in die Datenbank SQLBase ein, dabei werden wichtige Begriffe, Sachverhalte und Konzepte vorgestellt und erläutert; dieses Kapitel hat im Gegensatz zu den folgenden Kapiteln eher theoretischen Charakter;

Kapitel 2: stellt die Software vor, die für die Arbeit mit SQLBase in diesem Buch (und in der Regel auch später bei der Arbeit mit SQLBase) verwendet wird: SQLTalk, SQLConsole und Centura Team/Web Developer (dabei liegt der Schwerpunkt bezüglich der Sprache SQL in SQLTalk, SQLConsole wird mehr unter „administrativer“ Sicht vorgestellt, d. h. bei Verwaltungsaufgaben, wie z. B. dem Anlegen neuer Benutzer);

Kapitel 3: enthält alle SQL-Anweisungen sowie viele Beispiele hierzu und führt umfassend in diese Sprache ein;

Kapitel 4: enthält vertiefende Kapitel zur SQLBase, wie z. B. Partitionierung von Datenbanken, Query-Optimierung und ODBC/OLE DB Verwendung;

Kapitel 5: beinhaltet den Anhang mit Systemtabellen; außerdem werden die Tabellen der Beispiel-Datenbank ISLAND.DBS kurz vorgestellt, die in diesem Buch üblicherweise für Beispiele verwendet werden, sowie einen Index und eine Literaturliste.

Idealerweise hat man ab Kapitel 2 das Buch neben dem Computer liegen und übt dabei den Umgang mit der SQLBase.

Anmerkungen zur Literatur

Dieses Buch steht vom Inhalt her zwischen drei (englischsprachigen) Büchern, die von Centura herausgegeben werden:

- SQL Language Reference (einem einführenden Buch);
- Database Administrator's Guide (einem Buch für den Datenbank-Administrator) und
- Advanced Topics Guide (einem Handbuch für fortgeschrittene Anwender).

In einigen wenigen Fällen wird zur Vertiefung auf diese Bücher verwiesen. Andere Bücher, vor allem zum Thema SQL, finden sich im Anhang. Hinweise zu Centura-Produkten finden sich auch im Internet unter http://www.centurasoft.com. Auch die Suche im Internet unter Verwendung von Suchmaschinen kann Informationen zu Datenbanken liefern, so z. B. die Suche nach Transaktionen unter Verwendung von www.metacrawler.de, in der unter anderem der folgende Link auf einen umfangreichen Aufsatz angezeigt wurde:

http://medoc.offis.uni-oldenburg.de/Samples/rahmfree/mrdbs-5.html

Weitere Bücher die in Zusammenhang mit den hier behandelten Themen stehen (auch deutschsprachige), werden in der Literaturliste aufgeführt. Schließlich finden sich die genannten Centura-Bücher auch auf der SQLBase-CD (sie lassen sich über die Programmgruppe Centura/Centura Books Online mit dem Acrobat Reader lesen und - weitaus interessanter - nach Stichworten durchsuchen, s. beispielsweise DBA.PDF mit dem Inhalt des Buches "Database Administrator's Guide").

Außerdem finden sich auf der Homepage von Cenutra Hinweise zur Fehlerbehebung der Software, sog. PTF-Dateien (Program Temporary Fixes), so z.B mit dem Titel "Defects Corrected in SQLBase 6.1.0 PTF 4":

http://www.centurasoft.com/support/updates/temp_fix/supsb1p4.htm

SQL (Structured Query Language)

Da der Begriff SQL (Structured Query Language) an so vielen Stellen im Buch genannt wird und der eine oder andere Leser ggf. nicht das Buch von vorne bis hinten durcharbeitet, sei dieser Begriff hiermit an dieser Stelle eingeführt. SQL steht für eine Abfragesprache, die mit einem relativ kleinen Sprachschatz nicht nur alle Verwaltungsarbeiten ermöglicht, sondern darüber hinaus auch noch alle Werkzeuge enthält, die zum Eingeben, Ändern, Löschen oder Selektieren von Daten nötig sind. In diesem Buch findet sich ein umfangreiches Kapitel zu SQL.

Anmerkungen zur Syntax und eine Warnung

Bereits hier soll auf zwei wichtige Punkte hingewiesen werden:

- Immer wieder wird in diesem Buch (wie überhaupt in allen Büchern zum Thema SQL) eine SQL-Anweisung in der folgenden Form auftauchen:

  ```
  select * from company;
  ```

  Mit dieser Anweisung sollen aus einer Datenbank-Tabelle mit dem Namen COMPANY (sie befindet sich in der mitgelieferten Beispiel-Datenbank ISLAND.DBS) alle Datensätze gelesen und deren Inhalte angezeigt werden. Diese einfache Formulierung, die den Stern, der eigentlich Asterisk heißt, verwendet, um alle Spalten aus der Tabelle anzeigen zu lassen, wird allerdings nur der Einfachheit halber genutzt; diese Anweisung hat in einer „richtigen" Anwendung nichts verloren, dort würde sie in der folgenden Form auftauchen (man verwendet anstelle des Asterisks die tatsächlichen Spaltennamen):

```
select company_id, company_name, address, city
from company;
```

Weshalb das so ist? Man sollte auf den Asterisk „*“ immer verzichten, da er einen - eigentlich überflüssigen - Lesevorgang (s. Stichwort I/O) verursacht: um herauszufinden, welche Spalten in der Tabelle COMPANY überhaupt enthalten sind, muss die Datenbank zunächst in einer sog. Systemtabelle (hier: SYSCOLUMNS) nachschauen (dort wird über alle Spalten aller Tabellen „Buch geführt“); erst nachdem die Spalten ermittelt wurden, beginnt mit einem zweiten Lesevorgang das Lesen der eigentlich gesuchten Daten. Nicht immer wird die Wirkung messbar sein, aber man sollte es sich von Anfang an in der gezeigten Form angewöhnen (natürlich ist es mühsam, lange SQL-Anweisung immer wieder zu schreiben, aber man wird irgendwann dazu übergehen, in Anwendungen hierzu spezielle Variablen zu verwenden, in denen die Spaltennamen der verwendeten Tabellen gespeichert sind).

- Ein weiterer wichtiger Punkt: Viele Entwickler schreiben ihre Programme so, dass diese direkt mit Tabellen arbeiten (so wie oben das kurze Beispiel mit der Tabelle COMPANY gezeigt hat); dies mag für kleine Lösungen durchaus Sinn machen, bei großen Lösungen aber (d. h. umfangreiche Anwendungen oder viele Anwender - oder beides) ist es besser, für die Codierung ausschließlich Views (zu deutsch: Abfragen) zu verwenden und die Tabellen demjenigen zu überlassen, der sie erzeugt hat und auch pflegt: dem System-Administrator (SYSADM) bzw. dem Datenbank-Administrator (DBA). Hierzu findet sich noch ein spezielles Kapitel gegen Ende des Buches (s. Schema-Architektur, S. 273).

Geschwindigkeit

Immer wieder werden Hinweise folgen - so wie der eben genannte - die darauf abzielen, eine Anwendung nicht unnötig langsam zu machen. Je mehr Datensätze in einer Tabelle stehen, je mehr Anwender darauf zugreifen, je schlechter die verwendete Hardware ist, um so schneller werden Anwender den Administrator darauf aufmerksam machen, dass früher alles besser gewesen sei. Die in diesem Buch gemachten Hinweise sollen dazu führen, dass man sein konzeptionelles Vorgehen überdenkt, die SQL-Anweisungen optimiert und die Umgebung der SQLBase so einstellt, dass sie auch mit großen Datenmengen und hohen Anwenderzahlen gut zurecht kommt. Deshalb findet sich auch im Buch ein Kapitel, in dem erläutert wird, wie man eine langsam gewordene Anwendung wieder flott bekommt (Query Optimizer, S. 262).

SQLTalk

Eine gewichtige Rolle spielt in diesem Buch das Programm SQLTalk, mit dem alle SQL-Anweisungen und alle Verwaltungsaufgaben in Zusammenhang mit der Datenbank getestet und ausgeführt werden können. Außerdem bietet SQLTalk eine umfangreiche Hilfe mit allen Syntaxdiagrammen der einzelnen Befehle (die hier fast vollständig wiedergegeben sind). Die Syntaxdiagramme finden sich außerdem zusammen mit den Buchtexten in der Datei LANG.PDF, die in der SQLBase-CD enthalten ist.

Was fehlt?

Natürlich kann ein einführendes Buch nicht alle Aspekte bei der Arbeit mit SQLBase vorstellen und erläutern. Auf „Distributed Transactions“ (verteilte Transaktionen) wurde verzichtet. Zudem finden sich hier nur wenige Hinweise für Novell Netware-Anwender sondern eher für NT-Server Benutzer.

Inhaltsverzeichnis

1 Einstieg

In diesem Kapitel werden grundlegende Dinge beschrieben, die für das Verständnis von SQLBase notwendig sind. Ein Teil der hier gemachten Ausführungen wird dann in den späteren Kapiteln noch vertieft.

Was ist eine Datenbank, was ist eine Tabelle

Eine allgemein gehaltene Definition einer Datenbank könnte folgendermaßen lauten:

> Eine Datenbank ist eine Datei, die auf einem Datenträger gespeichert steht und in der Daten enthalten sind. Bei diesen Daten handelt es sich häufig um Daten, die geschäftliche Vorgänge beschreiben, und die beispielsweise anfallen, wenn etwas verkauft wird. Andere Datenbanken können Messwerte enthalten, die von einem Messwertgeber direkt oder indirekt in die Datenbank-Datei geschrieben werden.
>
> Um Daten speichern zu können, muss man in Datenbanken zuerst eine Struktur anfertigen, die Platz bietet für die unterschiedlichen Daten wie z. B. Kundennamen, Rechnungsnummern usw. Man bezeichnet diese Strukturen als Tabellen.
>
> Um mit einer Datenbank arbeiten zu können, benötigt man Software, zum einen um die Daten eingeben und abfragen zu können, zum anderen um die Datenbank-Anwender verwalten zu können.

Eine mehr an die Datenbank SQLBase angepasste Definition könnte so lauten:

> Eine SQLBase-Datenbank ist eine Sammlung von SQL-Objekten, wie z. B. Tabellen (für die Daten), Indizes (zur Beschleunigung von Abfragen u.a.m.) und Views (zur Auswahl einer Teilmenge der Daten). Eine Datenbank hat einen Dateinamen (wie z. B. ISLAND.DBS) und wird von einem Datenbank-Management-System geöffnet und verwaltet (bei der Fünf-Benutzer-Lizenz der SQLBase hat sie den Namen DBNT5SV.EXE). Die „Intelligenz“ dieses Datenbank-Management-Systems ist für die Geschwindigkeit und Stabilität der Datenbank verantwortlich (bei einfachen Systemen fehlt diese separate Software, vgl. die folgenden Ausführungen zu Client-Server-Datenbanken)

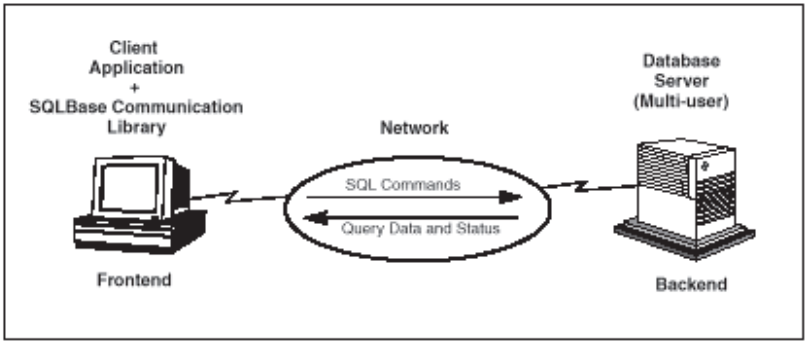
Client-Server-Datenbank und einfache Systeme

Wie kann man eine Client-Server-Datenbank wie beispielsweise SQLBase gegenüber einfacheren Datenbank-Systemen abgrenzen (einfachere Systeme werden oft als Flat-File-Systeme, Filebasierte Datenbanken bezeichnet).

Eine Client-Server-Datenbank besteht aus zwei Teilen: einer Datei, in der die Daten beheimatet sind, (hier die DBS-Dateien) und einer Software, dem sog. Datenbank-Management-System, das die Daten betreut und pflegt. Dieses Datenbank-Management-System wird von allen Anwendungen kontaktiert, die beim Anwender ausgeführt werden. Sollen also beispielsweise alle Kundendaten aus der Tabelle "Kunden" gelesen und auf dem Bildschirm eines Anwenders angezeigt werden, dann schickt der Client die hierzu nötige SQL-Anweisung an das Datenbank-Management-System, dort werden die Daten selektiert und „nach vorne“ zu dem Client

geschickt; (ein anderes Begriffspaar, das hierfür verwendet wird, lautet „Frontend“ und „Backend“).

Die folgende Abbildung stammt aus dem Band „Database Administrator's Guide“ und veranschaulicht diesen Zusammenhang



Dadurch, dass das Datenbank-Management-System mit einem gewissen Grad an „Intelligenz“ ausgestattet ist, werden Selektionen auf der Server-Seite bereits realisiert: möchte ein Anwender beispielsweise aus einer Menge von 100.000 Datensätzen nur fünf Datensätze selektieren, dann selektiert das Datenbank-Management-System genau diese fünf Sätze bereits auf dem Server und schickt genau diese fünf Sätze an den Anwender.

Ganz anders die einfacheren Systeme: dort holt sich der Client alle 100.000 Datensätze und selektiert die fünf gewünschten Sätze im eigenen Hauptspeicher (so macht das auch Microsoft Access!). Es liegt auf der Hand, dass dieses Verfahren für umfangreiche Datenmengen und/oder mehrere Anwender nicht vorteilhaft ist.

Abgrenzung zu anderen Datenbanken

Zur Abgrenzung von Datenbanken anderer Hersteller ist noch anzumerken, dass die Datenbank-Dateien untereinander (natürlich) nicht austauschbar sind: Will man Daten aus Oracle, Informix, Sybase, DB2, Ingres, SQLServer oder anderen am Markt vertretenen Datenbanken verarbeiten, gibt es dennoch mehrere Möglichkeiten: Importieren der Daten (z. B. über LOAD-Anweisungen, s. S. 16) oder Zugriff über JDBC-, ODBC-Treiber bzw. Verwendung der Schnittstelle OLE DB (hierzu folgen unten weitere Ausführungen). Auch sog. native Schnittstellen für die Verwendung aus einer SQLWindows-Anwendung heraus im Zugriff auf Datenbanken anderer Hersteller sind erhältlich (s. u. Centura Exchange).

Eigenschaften von Centura SQLBase

SQLBase (1984 in der ersten Version von Gupta, jetzt unter dem Namen Centura) war die erste relationale Client-Server-Datenbank, die speziell für PC-Umgebungen entwickelt wurde. Sie stand früher im direkten Gegensatz zu den Großrechner-Datenbanken oder jenen der mittleren Datentechnik. Die Zahl der Anwender, die sie heute verwenden, schwankt zwischen einer Million (Centura, Info Blatt) und zwei Millionen (COOPER). Dadurch, dass die Datenbank speziell für PCs entworfen wurde, war und ist es für sie einfacher „nach oben“ zu wachsen (d. h. in

Richtung stärkerer Hardware, mehr Anwender) als für andere Datenbanken, die zunächst für Großrechner entwickelt wurden und nun versuchen „nach unten“ zu den PCs bzw. den PC basierten Netzwerken vorzudringen. SQLBase ist besonders geeignet für Unternehmen, die mobile Mitarbeiter mit Datenbanken ausstatten müssen, reicht dabei vom Laptop-PC bis zur Mainframe (und sogar bis hinunter zur Scheckkarten-eingebetteten Datenbank); sie ist anerkanntermassen schnell und sicher, insbesondere die Version 7.x bietet durch neuartige Verschlüsselungstechnik eine sehr hohe Datensicherheit.

Datenbanken nach der Installation von SQLBase

Nach der Installation der SQLBase befinden sich eine Reihe von Datenbanken auf dem Ziel-Rechner:

START.DBS	Diese Datei befindet sich im Verzeichnis \Centura\ (das ist das vorgeschlagene Verzeichnis bei der Installation) und wird als Vorlage für neue Datenbanken verwendet. Wird eine neue Datenbank erstellt, passiert folgendes: 1. es wird ein neues Verzeichnis mit dem Namen der Datenbank erstellt; 2. die Datei START.DBS wird in das Verzeichnis hineinkopiert und umbenannt (sie erhält den Namen der Datenbank-Datei).
COUNTRY.DBS	Diese Datenbank wird benötigt, um eine an Landessprachen angepasste Version von SQLBase zu konfigurieren (das ist generell nicht erforderlich, kann aber in Einzelfällen, wo z. B. eine von der Norm abweichende Sortierung von Daten erwünscht ist, weiterhelfen, z. B. bei einer anderen Einordnung der deutschen Umlaute in den Zeichensatz; s. Administrator’s Guide Kap. 11).
ISLAND.DBS	Diese Beispieldatenbank wird in diesem Buch häufig verwendet, um die Syntax der Sprache SQL oder Mechanismen der SQLBase vorzustellen (sie enthält kaufmännische Daten).
DEMO.DBS	diese Datenbank kann unter Verwendung der Skript-Datei WINTUTOR.WTS angefertigt werden (in ihr befinden sich Daten zu US-Amerikanischen Präsidenten, s. S. 79).
MAIN.DBS	Diese Datenbank wird für partitionierte Datenbanken gebraucht und enthält Angaben z. B. über die Größe der Speicherbereiche und die Orte, an denen die einzelnen Partitionen abgelegt sind (eine Datenbank muss dann partitioniert werden, wenn sie unter NT die Größe von 512 GB und unter Netware die Größe von 26 GB überschreitet).

SQL.INI

Die Datei SQL.INI ist eine Datei mit Einstellungen zur Konfiguration der SQLBase. Ihre Inhalte werden beim Start der SQLBase gelesen. Da diese Datei wichtig ist, findet sich unten (S. 66) ein ausführliches Kapitel.

Begriffe von Centura

Einige Begriffe von Centura werden immer wieder in die Diskussion eingebracht, deshalb soll die folgende Übersicht etwas Klarheit schaffen:

Begriff	Erläuterung	Verwendung
CTD	Centura Team Developer	Software-Entwicklung
CWD	Centura Web Developer	Software-Entwicklung für Web-Anwendungen
Net.db	steht zwischen Database-Server und Web-Server (in der Business-Edition bedient er außerdem noch den Business Logic-Server)	zwei Arten: Standard und Business-Edition, erlaubt das Veröffentlichen von Datenbank-Daten (auch anderer Hersteller) im Web per Mausklick
SQLBase	Server-Datenbank, relationales Datenbank-Management-System	stellt Daten für Clients zur Verfügung
SQLBase 7.5	die eigentliche Datenbanksoftware;	relationales Datenbank Management
Safegarde	Safegarde deutet auf die Möglichkeiten der Datenverschlüsselung hin	System (RDBMS)
SQLTalk	Programm zur Wartung von Datenbanken (DBS-Dateien); wird in diesem Buch vorwiegend verwendet	da mit diesem Programm alle (!) SQL-Anweisungen ausführbar sind, lassen sich auch alle Arbeiten an der Datenbank damit ausführen
Quest	früher verwendetes Client-Werkzeug	zum Erstellen von Tabellen, Abfragen, Reports etc.
SQLBase Exchange	Software für Replikation von Datenbanken	ermöglicht Replikation, Import/Export von Daten in Fremdformaten, unterstützt: Oracle, Sybase, Informix, DB2, SQLServer, ODBC, Access, dBASE
SQLHost	Integration von DB2-Datenbanken	Zugriff auf DB2-Daten auch aus CTD/CWB heraus
SAL	Scalable Application Language	Programmiersprache für Centura Team/Web Developer
OLE DB	Schnittstelle zwischen Datenbank und Anwendersoftware	wird ODBC (Open database Connectivity) ablösen, ist auch für objektorientierte Datenbanken u.a. zu verwenden

1.1 Anmerkungen zur Installation und erste Tätigkeiten

Hier sollen einige wichtige Begriffe erläutert werden, die mit der Installation der SQLBase zusammenhängen, aber auch solche, die im Betrieb der SQLBase immer wieder verwendet werden.

1.1.1 Verzeichnis

Als Default-Verzeichnis für die Installation wird immer das Verzeichnis (oder: der Ordner) „\Centura“ verwendet; das heißt, eine neue Datenbank wird beispielsweise in dem Verzeichnis „\Centura\MyData“ abgelegt. Natürlich ist es auch möglich, das Verzeichnis anders festzulegen, so dass z. B. eine Datenbank „\MyData“ direkt unter dem Root-Verzeichnis abgelegt wird, hier wird jedoch von dem Standard-Verzeichnis ausgegangen.

Jede Datenbank, die angelegt wird, erhält ein eigenes Verzeichnis unter diesem vorgegebenen Verzeichnis „\Centura“. So könnte z. B. die folgende Verzeichnis-Struktur entstehen (hier auf dem Server-Laufwerk K:, unter dem Verzeichnis Centura):

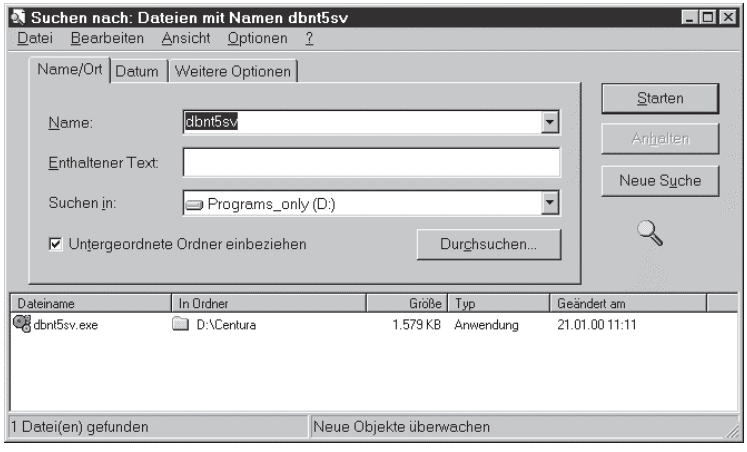
K:\Centura
K:\Centura\Auftrag
K:\Centura\MaSurvey

1.1.2 Start, Stop des Servers

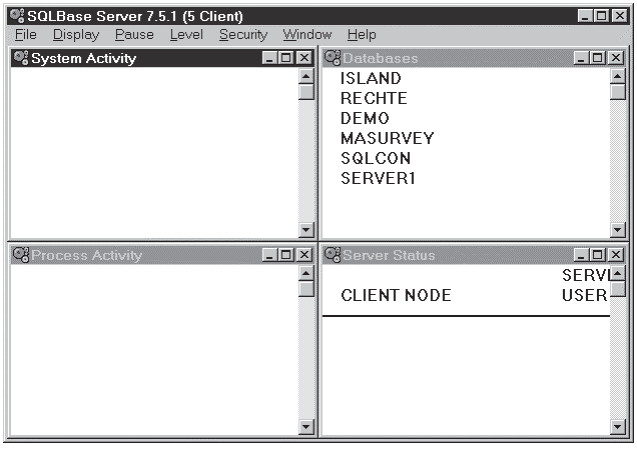
Spricht man vom Starten, meint man üblicherweise, dass man die SQLBase startet. Häufig wird dann aber erwartet, dass Datenbanken ebenfalls schon verfügbar sind. Das ist aber nicht ganz richtig: Zwar zeigt die SQLBase bereits Datenbank-Namen an (nämlich solche, die in SQLBase „bekannt“ sind - bekannt werden sie durch einen Eintrag in der SQL.INI), arbeiten kann man aber mit den Daten erst, nachdem (1) eine Client-Anwendung wie z. B. SQLTalk gestartet wurde und (2) dort mittels der Anweisung „connect island;“ die Datenbank geöffnet wurde.

start

Bevor mit einer Datenbank gearbeitet werden kann, muss der Server (genaugenommen die SQLBase) gestartet werden. Wie schon erwähnt, gibt es diverse Versionen der SQLBase. Hier soll die fünf-User-Version unter Windows NT gestartet werden, die den Namen „dbnt5sv.exe“ hat.



Da es sich um eine ausführbare EXE-Datei handelt, genügt es durchaus, den Dateinamen doppelt anzuklicken. Nach dem Doppelklick erscheint ein Fenster, in dem über den Menüpunkt Display/All die folgenden Informationen angezeigt werden:



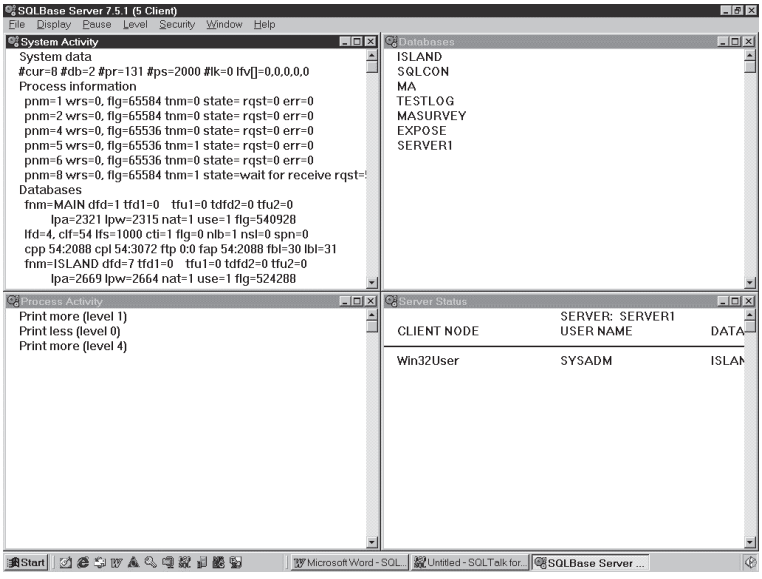
In der vorstehenden Abbildung oben rechts finden sich die Namen der in der Datei SQL.INI eingetragenen Datenbanken (das sind die der SQLBase bekannten, aber zu diesem Zeitpunkt noch nicht geöffneten Datenbanken); an den beiden leeren Fenstern links ist erkennbar, dass noch kein User eingeloggt ist.

Natürlich gibt es noch andere Möglichkeiten, die SQLBase zu starten:

- über Start/Programme/Centura/ SQLBase 7.5.1;
- innerhalb von SQLTalk durch die Anweisung „set server server1;“ (falls sich „server1“ auf dem lokalen Rechner befindet);
- durch den Start von SQLConsole (s. S. 91);
- durch den Start von Centura Team/Web Developer (dort klickt man dann auf den Menüpunkt Database Explorer);
- unter Verwendung einer API-Funktion z. B. aus C++ oder VB heraus;
- durch Verwendung des SQLBase Server Monitors SSM (s. S. 20)
- als NT-Service. (s. S. 17).

SQLBase, gestartet und bereit

Die folgende Abbildung zeigt die Server-Meldungen, nachdem zunächst die Datenbank ISLAND geöffnet und eine SELECT-Anweisung ausgeführt wurde (select * from company):



Um die Meldungen so umfangreich ausfallen zu lassen, wurde über den Menüpunkt Level auf Level 4 umgestellt, das wird im Fenster unten links entsprechend protokolliert „Print more (level 4)“.

Die Abkürzungen in den Meldungen links oben bedeuten:

- pnm-process number
- wrs-waiting for resource
- tnm-transaction number
- state= rqst-waiting for request
- rqs-letzter emfangener request
- S-shared lock (gesetzt nach Select-Anweisung), andere Locks z. B. X-Lock (Exclusive Lock z. B. nach Update-Anweisung; zu dem Thema Locks folgen unten weitere detaillierte Ausführungen s. S. 50)

Anzeigen von Daten aus der Datenbank Island mit SQLTalk

Nachdem die SQLBase über eine der vorstehend genannten Arten gestartet wurde, können Daten - beispielsweise aus der Datenbank Island - angezeigt werden, hier unter Verwendung von SQLTalk

Möglicherweise muss zunächst eine Verbindung zum Server hergestellt werden. Hierzu lässt man in SQLTalk die folgende Anweisung ausführen:

```
set server server1;
```

Danach wird mittels CONNECT auf die Datenbank zugegriffen. Mittels SELECT werden Daten gelesen und angezeigt (hier zunächst aus der Systemtabelle SYSTABLES, danach die Datensätze aus der Tabelle CATALOGS):

```
connect island;
select * from systables;
select * from catalogs;
```

Stoppen

Unter „Stoppen“ versteht man das Herunterfahren der SQLBase (üblicherweise nachdem sich die Anwender von der Datenbank abgemeldet haben):

1. Abmelden von der Datenbank

Mit der folgenden Anweisung meldet man sich innerhalb von SQLTalk von der Datenbank ab:

```
disconnect island;
```

Daraufhin meldet SQLTalk:

```
ALL CURSORS FOR DATABASE ISLAND DISCONNECTED
```

2. Stoppen Server

In **SqlTalk** unterbricht man die Verbindung zum Server (ohne ihn herunterzufahren) durch:

```
set server off;
```

Hierdurch wird der Eintrag „ClientNode“, „Win32User“ (s. Abbildung oben, Datenbank-Fenster) aus der Liste enfernt, außerdem meldet SQLTalk:.

```
SERVER IS OFF
```

1.1.3 Installieren und De-Installieren einer Datenbank

Unter dem Begriff „Installieren einer Datenbank“ versteht man, dass die Datenbank (die bereits vorhanden ist und möglicherweise auch schon Daten enthält) der SQLBase bekannt gemacht wird. Hierzu muss der Datenbankname und das zu verwendende Protokoll (im folgenden „sqlapipe“) in der Datei SQL.INI eingetragen werden (manuell oder unter Verwendung von SQLConsole oder des SSM-System Service Managers).

Der Name der Datenbank wird in der Datei SQL.INI nach dem Schlüsselwort „dbname“ aufgelistet (ggf. zusammen mit den anderen installierten Datenbanken, z. B. in folgender Form, hier für die 1-User-SQLBase:

```
[dbnt1sv]
dbdir=D:\Centura
dbname=ISLAND,SQLAPIPE
dbname=MA,SQLAPIPE
```

Achtung: Es kann sehr leicht passieren, dass man sich in der Sektion irrt, analog zu der hier genannten [dbnt1sv] gibt es natürlich auch die Sektion [dbnt5sv].

Beim De-Installieren wird genau das Gegenteil gemacht, d. h. der Eintrag der Datenbank wird aus der Datei SQL.INI entfernt (die Datenbank wird dadurch natürlich nicht gelöscht); die darin befindlichen Daten sind allerdings dann nicht mehr verfügbar.

In **SQLTalk** verfährt man wie oben geschildert: man verwendet einen Text-Editor und ändert den Eintrag in der SQL.INI (am besten deaktiviert und kommentiert man den Eintrag durch ein

führendes Semikolon deaktivieren, so wie im folgenden Beispiel für die Datenbank ISLAND gezeigt):

```
[dbnt1sv]
dbdir=D:\Centura
;deaktiviert am 11.11.1999
;dbname=ISLAND,SQLAPIPE
dbname=MA,SQLAPIPE
```

1.1.4 Datenbanken erzeugen und löschen: create und drop

Mit create bzw. drop meint man das Anfertigen bzw. Löschen von Datenbanken. Da es sich um normale Dateien mit der Extension DBS handelt (wenn man einen Augenblick vergisst, dass bei partitionierten Datenbanken mehrere DBS-Dateien auf verschiedene Laufwerke verteilt sind), dann ist es einleuchtend, dass man zum Löschen auch den Explorer verwenden kann. Man sollte dies aber überhaupt nicht tun und diese durchaus sensiblen Operationen unter Verwendung von SQLTalk oder SQLConsole erledigen.

create

Man kann eine Datenbank **manuell** (auf der „DOS-Ebene“ oder mit dem Explorer) erstellen, indem man folgendes macht:

- *Erstellen* eines neuen Ordners (hier unter dem bereits bestehenden Ordner \CENTURA, anderen Namen wählen, falls er festgelegt wurde), beispielsweise mit dem Namen TEST (keine Extension),
- *Kopieren* der Datei START.DBS in das neue Unterverzeichnis (hier: \Centura\Test),
- *Umbenennen* der Datei auf den neuen Datenbank-Namen (hier: Test.DBS),
- *Eintragen* des Namens in die Datei SQL.INI (in der entsprechenden Sektion, abhängig von der verwendeten Datenbank).

Eine Datenbank mit **SQLTalk** fertigt man wie folgt an:

- Man startet zunächst SQLTalk, stellt dort ggf. eine Verbindung zum Server her:
  `set server server1;`
- und schickt dann die SQL-Anweisung „create database ...“ ab;
  `create database test;`

Das zuletzt geschilderte Verfahren ist sicher der Normalfall, aber es ist zu Testzwecken interessant, auch einmal eine DBS-Datei manuell zu erstellen.

drop

Um eine Datenbank mit SQLTalk zu löschen, verfährt man ähnlich,

```
drop database test;
```

Nochmals: Man sollte äußerst vorsichtig mit der DROP-Anweisung umgehen. Im Zweifel macht man zuerst eine Sicherungskopie der DBS-Datei zusammen (!) mit den LOG-Dateien (am einfachsten innerhalb von SQLConsole: Mausklick rechts auf Datenbank, Backup/Snapshot).

1.1.5 Unload, Load

Unter Unload/Load versteht man das Exportieren von Datenbank-Daten in ein Fremdformat oder das Importieren von Daten aus einem Fremdformat. Das Kommando ist außerordentlich vielseitig, da z. B. verschiedene Formate unterstützt werden; es wird deshalb in einem umfassenden Kapitel separat vorgestellt und erläutert. Hier nur soviel: Man verwendet die Unload-Anweisung aus zwei Gründen: um **Datenaustausch** mit anderen Datenbanken zu ermöglichen oder um **Sicherungskopien** herzustellen (letzteres tritt hierbei „in Konkurrenz“ mit den „Backup“-Anweisungen, die noch häufiger als die Unload-Anweisung verwendet werden).

1.1.6 Connect, Disconnect

Die Anweisung Connect taucht in zweierlei Formen auf:

- CONNECT als SQL-Anweisung, um eine Datenbank zu öffnen, so z. B. in SQLTalk:
  ```
  connect island;
  ```
- CONNECT (neben RESOURCE und DBA als Zuweisung von gewissen Rechten (oder Privilegien) an diverse Benutzer, innerhalb von SQLTalk:
  ```
  grant connect to Mary;
  ```

Zu beiden Varianten folgen unten ausführliche Beispiele.

1.1.7 Update von SQLBase (unload, load, shutdown)

Da bei der Weiterentwicklung der SQLBase nicht nur die Programmlogik weiterentwickelt wird, sondern auch das Format, in dem die Daten gespeichert werden, muss jede SQLBase-Version separat betrachtet werden. Zwar ist es bei manchen Versionen möglich, das Format automatisch zu aktualisieren (z. B. von Version 6.1.x auf Version 7), aber einige Sicherheitsvorkehrungen wollen dennoch beachtet werden: dazu gehört das bereits oben genannte Unload der Datenbank, die auf die neue SQLBase-Version „verschoben“ werden soll.

Das heißt, man darf auf keinen Fall eine neue SQLBase-Version über eine vorhandene Version installieren. Typischerweise muss man vor einer Installation die Daten der Datenbanken (also der DBS-Dateien) zuerst exportieren, dann die neue SQLBase installieren, um anschließend die Daten wieder zu importieren. Ein Beispiel soll das Verfahren zeigen.

Beispiel zum Update der SQLBase-Software auf Version 7

In der folgenden Übersicht wird vergleichend aufgeführt, wie ein Update auf die Version 7 zu machen ist. Es wird dabei differenziert nach den SQLBase-Versionen **6.0.x** und den Versionen **6.1.x**, dabei kann festgehalten werden, dass sich die beiden Verfahren nur in der fett formatierten Zeile unterscheiden (wie man von den älteren SQLBase-Versionen 5.x ein Update macht, ist außerdem in der Datei RELEASE.HTM auf der SQLBase-CD beschrieben):

Update von Version 6.0.x	Update von Version 6.1.x
unload der DB-Dateien	unload der DB-Dateien
Sichern der SQL.INI	Sichern der SQL.INI
shutdown des Servers	shutdown des Servers
Installieren des neuen Servers, starten	Installieren des neuen Servers, starten
DB-Create, load der DB-Dateien	- keine Aktion -
connect	connect

Beispiel

Die auf der Homepage des Autors (s. Vorwort) befindliche Datei EBI.ZIP enthält neben dem Source-Code für eine Auftragsabwicklung auch die Datei EBI.ULD. Diese Unload-Datei wurde aus einer DBS-Datei unter der SQLBase-Version 5 erstellt und wird im folgenden Beispiel in SQLBase Version 7.x geladen. Die Reihenfolge der Anweisungen in SQLTalk:

Alte SQLBase-Version zum Erstellen der Unload-Datei:

```
connect ebi;
unload database ebi.uld;
```

Die ULD-Datei enthält alle nötigen Daten zum Erstellen einer neuen DBS-Datei unter einer neuen SQLBase-Version. Sie wird zweckmässigerweise mittels Winzip komprimiert und als Archiv abgelegt.

Neue SQLBase-Version zum Erstellen einer neuen DBS-Datei:

```
create database ebi;
connect ebi;
load sql K:\centura\sicherung\ebi.uld;
```

Damit sind die Daten in der neuen SQLBase-Version verfügbar.

Skript-Datei

Man kann auch eine Datenbank komplett neu mit einem Skript erstellen lassen (from scratch). Hierzu muss eine solche Skript-Datei natürlich vorliegen, in der alle hierzu nötigen SQL-Anweisungen und ggf. die Daten enthalten sind. Diese Skript-Datei kann man in SQLTalk laden und ausführen lassen.

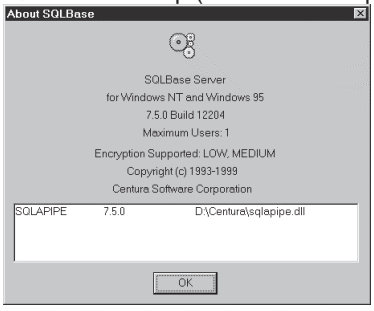
Eine in diesem Buch vorgestellte Skript-Datei hat den Namen WINTUTOR.WTS, sie wird auf S. 79 besprochen.

1.1.8 Packages, SSM, NT-Service, Sicherheits-Level, Neuigkeiten

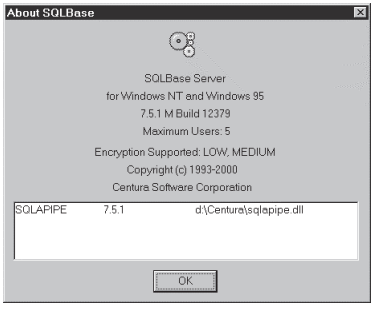
Packages

Bei der Installation der SQLBase wird von sog. Packages gesprochen. Zwei Typen werden unterschieden:

- **SQLBase Desktop** (das ist die Single-User-Lizenz): Es wird maximal *eine Verbindung* zur Datenbank gestattet, allerdings ist die Verbindung **Multi-Tasking**-fähig, d. h. mehrere Anwendungen können auf die gleiche Datenbank zugreifen (die dürfen ab Windows 95/98/NT auch remote liegen); Hinweis: Es ist möglich, die Single-User-Lizenz zweimal zu installieren und mit unterschiedlichen Konfigurationen zu starten; hierzu muss der Registry-Eintrag HKEY_LOCAL_MACHINE geändert werden; nähere Hinweise finden sich im Starter-Guide Kap. 4-3 der SQLBase; die folgende Abbildung zeigt das About-Fenster der Single-User-Lizenz bzw. SQLBase Desktop (über den Menüpunkt Help/About):



- **SQLBase Server** (das sind die Multi-User-Lizenzen): zu den **Multi-Tasking**-Fähigkeiten (s. vorstehend) kommen hier noch die **Multi-User**-Fähigkeiten hinzu (d. h. es kann eine zuvor festgelegte und lizenzierte Zahl von Anwendern gleichzeitig mit den Daten arbeiten; es gibt Lizenzen für 5-, 10-, 25-, 50 User sowie eine sog. unlimited Lizenz, in der die Zahl der gleichzeitigen User nicht beschränkt ist. Die folgende Abbildung zeigt das Fenster der 5-User-Lizenz:



SQLBase unter Windows NT: Application oder NT-Service

Installiert man die NT-Version der SQLBase, dann wird diese zunächst als sog. *Application* auf dem Server eingerichtet (also so wie jede andere Software auch). Man kann aber SQLBase auch als sog. *NT-Service* einrichten und ablaufen lassen. Das heißt, dass eine solchermaßen eingerichtete Anwendung *beim Start des Betriebssystems (NT) automatisch hochgefahren* werden kann ohne manuelles Zutun und ohne dass sich ein Anwender mit Connect an die Datenbank hängen müsste, was insgesamt die Administration vereinfacht (s. hierzu den Ordner Start/Einstellungen/Systemsteuerung/Dienste).

Allerdings sollte SQLBase nur dann als NT-Service eingerichtet werden, wenn die Hardware ausreichend schnell ist (und das heißt u.a.: die Taktrate muss größer sein als 133 MHz-

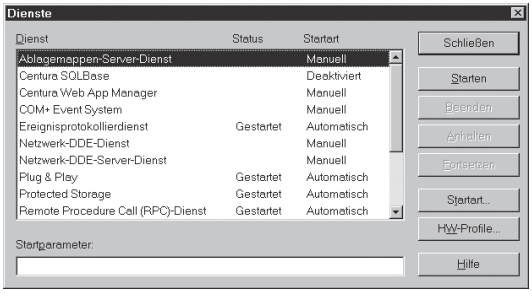
Pentium); andernfalls dauert das Starten so lange, dass Timeout-Fehler auftreten können (s. a. Datei RELEASE.HTM auf der Installations-CD der SQLBase).

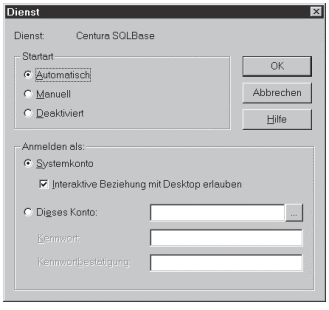
Vorteile und Eigenschaften von SQLBase als NT-Service:

- SQLBase wird automatisch beim Start des Servers hochgefahren, auch ohne dass ein Anwender sich einloggen müsste (dabei kann für den NT-Service als „Start-Art“ neben „automatisch“ auch „manuell“ festgelegt werden);
- es bestehen weniger Möglichkeiten, durch „Basteln“ an der SQLBase etwas zu ändern;
- Meldungen erscheinen direkt im Ereignis-Protokoll von NT (s. Programm EVENTVWR.EXE oder Start/Programme/Verwaltung (Allgemein)/Ereignisanzeige);
- zur Verwaltung der SQLBase als NT-Service wird entweder der NT-Service Manager oder der SQLBase Server Monitor (SSM) verwendet.

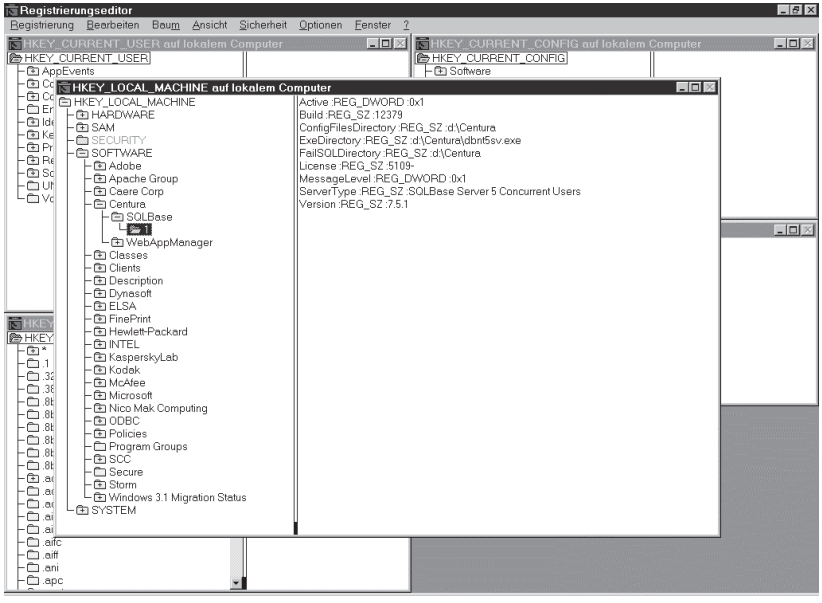
NT-Service Manager

Nach der Installation ist im NT-Service-Manager der Eintrag „SQLBase, Deaktiviert“ enthalten (linke Abbildung, zweite Zeile). Durch einen Doppelklick auf den Eintrag „Centura SQLBase“ öffnet sich das Fenster zum Konfigurieren des NT-Services (rechte Abbildung):



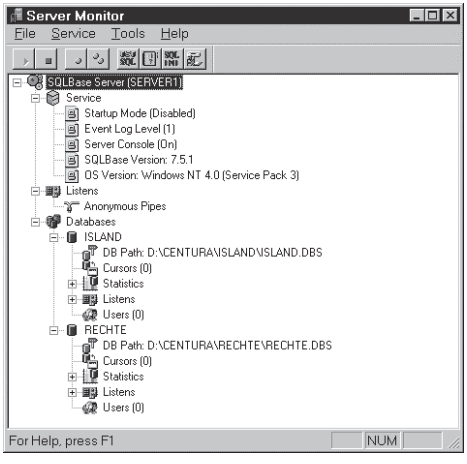


Auch die Eintragungen in der Registry von NT (die Restrierungsdatenbank, in der alle Service-Anwendungen aufgelistet sind) lassen sich anzeigen (hierzu muss das Programm REGEDT32.EXE gestartet werden):



SQLBase Server Monitor (SSM)

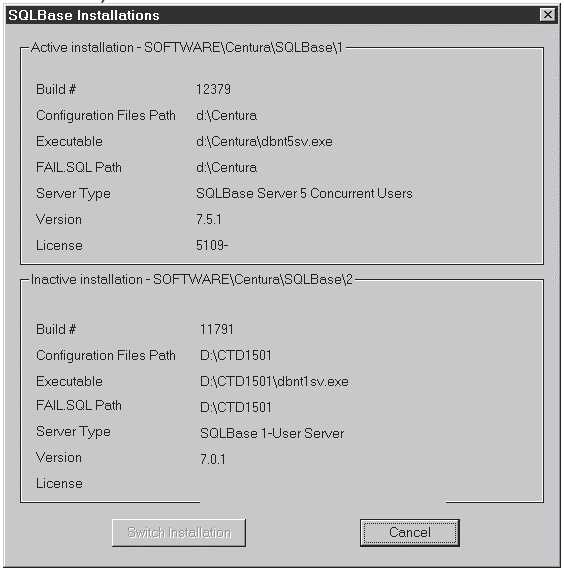
Der Server Monitor SSM (SSM.EXE) kann - bei gestarteter SQLBase - als Verwaltungswerkzeug (neben der etwas langsamer startenden SQLConsole) verwendet werden (das geht sogar dann, wenn SQLBase als Application Service eingerichtet wurde). Der SSM zeigt folgende Informationen zu den Datenbanken: Speicherort, Dateigröße, bei (partitionierten und unpartitionierten) Datenbanken den verfügbaren Speicherplatz, Netzwerkprotokolle und (!) die derzeit eingeloggten Anwender.



Am einfachsten lässt sich SQLBase starten, indem man in der vorstehenden Abbildung auf die Schaltfläche in der Symbolleiste (ganz links) klickt. Rechts daneben findet sich die Schaltfläche zum Stoppen der SQLBase.

Weitere Eigenschaften bzw. Möglichkeiten des SSM:

- Falls die SQLBase als Application eingerichtet ist (die Voreinstellung), lässt sie sich zwar nicht über SSM starten, aber wenn sie herkömmlich gestartet wurde, kann man SSM trotzdem verwenden (auch zum Herunterfahren der SQLBase);
- einfacher Start der SQLBase innerhalb des SSM über (1.) Service/Automatic und (2.) File/Start SQLBase Service, danach hat der SSM die Kontrolle über die SQLBase (es ist z. B. nicht mehr möglich, die SQLBase über Alt-F4 oder SQLTalk herunterzufahren, das muss jetzt innerhalb des SSM über den Menüpunkt File/Stop SQLBase geschehen);
- der Event-Viewer lässt sich innerhalb des SSM direkt anzeigen (über Menüpunkt Tools/Event Viewer);
- über einen Mausklick rechts auf die Schaltfläche „Installations“ lassen sich - falls vorhanden - zwei Installationen der SQLBase anzeigen (so wie in der folgenden Abbildung die 5-User-Lizenz oben und die Single-User-Lizenz unten), außerdem kann man über die Schaltfläche „Switch Installations“ die aktive Installation auf die jeweilige andere schalten (s. oben HKEY_LOCAL_MACHINE).

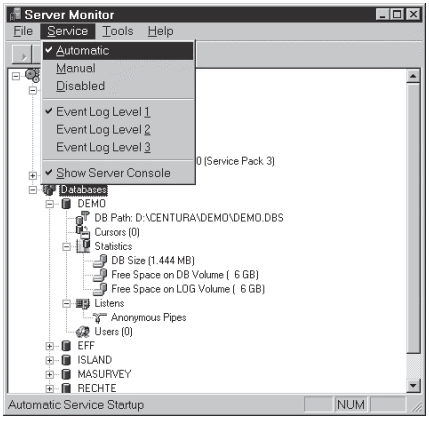


Einstellungen des SSM

Im SSM kann über den Menüpunkt Service festgelegt werden, wie SQLBase gestartet werden soll. Drei Möglichkeiten stehen zur Auswahl:

- Automatic: beim Start von Windows-NT wird SQLBase automatisch mitgestartet;
- Manual: nachdem NT gestartet ist, kann SQLBase manuell gestartet werden;
- Disabled: SQLBase kann nicht als NT-Service gestartet werden (nur herkömmlich als Application).

Die folgende Abbildung zeigt die entsprechenden Möglichkeiten.

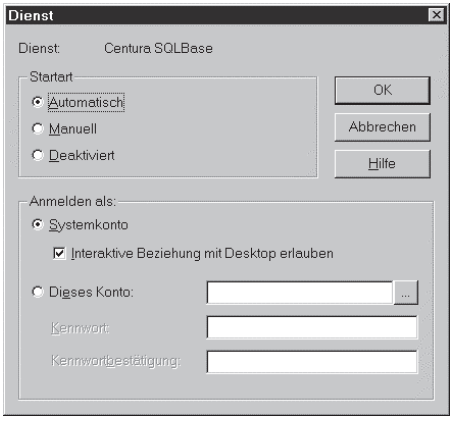


Alle Einstellungen werden erst nach einem Neustart der SQLBase wirksam.

Wechsel der Startart: SQLBase als NT-Service einrichten

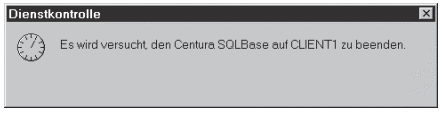
Bei der Installation von SQLBase wird die Software automatisch als Application eingerichtet (nicht veränderbare Voreinstellung; es ist nicht möglich, SQLBase „sowohl als auch“, d. h. als Application *und* als Service, einzurichten). Man kann die SQLBase aber nachträglich als NT-Service einrichten, und das geht folgendermaßen:

- im Ordner Start/Einstellungen/Systemsteuerung/Dienste den Eintrag SQLBase markieren,
- Startart... anklicken, im folgenden Fenster auf „Automatisch“ umstellen (s. Abbildung);



- nach Schließen des Fensters und Neustart des Rechners wird von nun an die SQLBase automatisch hochgefahren.

Wichtig: Ist die SQLBase als NT-Service eingerichtet, lässt sie sich nicht mehr mit Alt-F4 beenden, sondern muss entweder innerhalb des SSM heruntergefahren werden (Knopf in der Schaltfläche) oder aber über Start/Einstellungen/Systemsteuerung/Dienste, Schaltfläche „Beenden“; NT meldet dann das Herunterfahren:



Neuigkeiten in Version 7

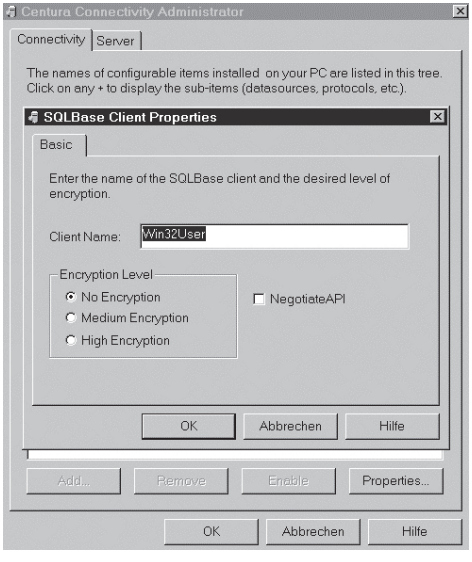
Die SQLBase-Version 7 bietet einige Neuigkeiten, die hier kurz aufgelistet werden sollen:

- Mit anderen Entwicklungswerkzeugen (z. B. C, C++) erstellte, sog. externe Funktionen (die z. B. in DLL-Dateien gespeichert sind) lassen sich nach entsprechender Deklaration in der Datenbank verwenden;
- mehrfache (multiple) Transaktionen sind in der Datenbank möglich;
- unter Windows NT kann SQLBase jetzt auch als sog. NT-Service laufen (s. vorstehende Ausführungen);
- der neue SQLBase-Server-Monitor hilft bei der Online-Verwaltung von SQLBase, beispielsweise um herauszufinden, wo Abfragen zu langsam ausgeführt werden;
- der Connectivity Administrator hilft bei der Konfiguration der SQLBase;
- ODBC (Open Database Connectivity Version 3) –Treiber erlauben den Zugriff von anderen Anwendungen auf die Daten in den Tabellen (so beispielsweise. für Programme, die mit Visual Basic oder C++ erstellt wurden).

Sicherheits-Level

SQLBase ist auf verschiedenen Sicherheits-Levels konfigurier- und einsetzbar. Um die Einstellungen einzusehen bzw. zu ändern startet man den Connectivity Administrator (nicht zu verwechseln mit dem Connectivity Manager innerhalb von SQLConsole) über Start/Programme/Centura/SQLBase7.5.1/ Connectivity Administrator.

Dort klickt man doppelt auf den Eintrag SQLBase und erhält folgendes Fenster angezeigt:



Hier lässt sich der Grad der Verschlüsselung festlegen (Encryption Level). Zur Wahl stehen:

- **SQLBase Standard**: verhindert „passwort guessing“, also das Erraten von Passworten;
- **SQLBase SafeGarde**: enthält die Verschlüsselungsstufen LOW und MEDIUM; die Page-Verschlüsselung auf dem Datenträger verhindert auch das Lesen der Daten bei Benutzung anderer Software; verhindert ferner das Lesen von Daten, die z. B. über das Netzwerk-Kabel übertragen werden;
- **SQLBase SafeGarde Max**: enthält alle genannten Eigenschaften und zusätzlich die Verschlüselungsstufe HIGH (128 bit triple DES).

Weitere Hinweise finden sich unter der Anweisung ALTER DBSECURITY,S. 173)

1.1.9 SYSADM und DBA

Zwei Kürzel tauchen bei der Arbeit mit SQLBase immer wieder auf:

- **SYSADM** (System Administrator), der eigentliche „Chef“ der Datenbank (es gibt immer nur einen Chef je Datenbank). Er steht - betrachtet man die Befugnisse - an erster Stelle in der Hierarchie, er hat universale Rechte.
- **DBA** (Database Administrator): Anwender, die mit diesem Recht ausgestattet sind (sie erhalten sie vom SYSADM) stehen in der Hierarchie unter dem SYSADM, haben noch immer sehr weitreichende Rechte, sind aber nicht so „mächtig“ wie der SYSADM. Es kann mehrere Anwender mit DBA-Befugnissen geben (in großen Netzwerken), es kann aber auch sein, dass der SYSADM gleichzeitig die Funktion des DBA wahrnimmt (in kleineren Netzwerken).

Zu den Aufgaben des SYSADM/DBA zählen u.a. die folgenden

- Ausstatten von Anwendern mit Zugriffsrechten für die einzelnen Tabellen;
- Erstellen von Indizes;
- Prüfen der Datenbank-Performance, ggf. dafür sorgen, dass die Performance (z. B. durch Anfertigen neuer Indizes oder Löschen nicht benötigter Indizes) gesteigert wird (hierzu wird intensiv Gebrauch von dem Query Optimizer gemacht (s. S. 262);
- regelmässige Backup-Dienste zur Sicherung der Daten;
- Erstellen von Views, die auf einzelne Benutzer(-gruppen) zugeschnitten sind;
- Auslagern von Programmcode in Stored Procedures, Erstellen von Triggern.

All die genannten Tätigkeiten werden in diesem Buch erläutert.

1.2 Über Tabellen

1.2.1 Primary Key, Foreign Key, Relation

Oben wurde bereits erwähnt, dass die Tabellen die „Heimat“ der Daten sind. Tabellen können nach ihrem **Zweck** differenziert werden:

- **Tabellen mit Daten**, die für eine Anwendung gespeichert sind (z. B. Tabellen mit Kundendaten, Lagerdaten usw.) und
- Tabellen, die für den Betrieb der Datenbank notwendig sind, sog. **Systemtabellen**; (z. B. wird das Recht eines einzelnen Anwenders, eine bestimmte Tabelle öffnen zu dürfen, in einer solchen Systemtabelle gespeichert).

Von den selbst erstellten Tabellen kann man weiter nach dem darin stehenden **Inhalt** differenzieren:

- Tabellen mit **Parent**-Daten, beispielsweise Kunden-, Personal-, Artikel-, Lieferanten-Stamm-Daten; andere Begriffe hierfür sind Master- oder Stammdaten;
- Tabellen mit **Child**-Daten beispielsweise solche, in denen Daten einzelner Aufträge gespeichert sind; (ein anderer Begriff hierfür ist Detaildaten).

Es gibt weitere Begriffspaare für die Master-/Detailbeziehungen:

- Master/Detail
- Parent/Dependant
- Parent/Child

Welcher der Begriffe verwendet wird, ist unerheblich, in diesem Buch ist überwiegend von Parent und Child die Rede.

Allerdings soll gleich hier darauf hingewiesen werden, dass die „Rollen“ bezüglich Parent und Child nicht fixiert sind: Eine Tabelle, die in einer ersten Beziehung die Parent-Rolle spielt, kann in einer anderen Beziehung durchaus die Child-Rolle übernehmen. Zum Beispiel die Tabelle INVOICE aus der Datenbank Island: sie enthält Auftrags-Masterdaten, also allgemeine Eigenschaften eines Auftrags, s. Anhang S. 303): in der Beziehung mit der Tabelle COMPANY hat sie die Rolle der Child-Tabelle (in der Master-Detail-Beziehung hat eine Company viele Aufträge); in der Beziehung mit der Tabelle INVOICE_ITEM (diese enthält die einzelnen im Auftrag abgegebenen Artikel) dagegen ist sie die Parent-Tabelle (ein Auftrag enthält üblicherweise viele Detailsätze).

Eine **Master-Tabelle** könnte beispielsweise folgendermaßen aufgebaut sein (hier anhand einer Kunden-Master-Tabelle, in der alle Eigenschaften der Kunden gespeichert werden); zur Demonstration sind hier nur einige wenige Felder - auch Spalten genannt - wiedergegeben:

KundenNummer	Firma	Abteilung	Ansprechpartner	Telefon
...				
4318	ABC Druck	Buchhaltung	Frau Meier	0123-456
4789	XXX Handel	EDV	Herr Keller	0321-987
...				

Üblicherweise erhält das Feld KundenNummer dieser Parent-Tabelle einen sog. Primärschlüssel (Primary Key), der den Feldinhalt eindeutig hält und für kürzere Zugriffszeiten sorgt (s. u. CREATE INDEX, S. 116).

Eine **Detail-Tabelle** dagegen könnte folgenden Aufbau und Inhalt haben (hier enthalten die Detaildaten kurze Notizen, die bei telefonischen Kontakten mit Kunden anfallen):

KundenNummer	Datum	Thema	Ergebnis
...			
4318	4.5.99	Anfrage Rabatt	nein
4318	7.8.99	Lieferung verspätet	Weiterleitung an Frau May
4318	1.9.99	Konferenz Malibu	ok
...			

In dieser Detail-Tabelle bezeichnet man das Feld KundenNummer als Foreign Key (Fremdschlüssel): die Feldinhalte werden mit dem gleichnamigen Feld der Parent-Tabelle, das dort einen Primary Key hat, verknüpft. In dieser Detail-Tabelle sind Wiederholungen normal (im Gegensatz zu den Daten der Parent-Tabelle, dort erscheint jeder Datensatz, also z. B. jeder Kunde, nur einmal).

Über das in beiden Tabellen vorkommende Feld „Kundennummer“ können die Tabellen miteinander verknüpft werden (dabei entsteht die oben erwähnte Relation). Diese Verknüpfung zwischen zwei Tabellen kann man grafisch folgendermaßen darstellen:

Hierbei steht die Master-Tabelle links, die Detail-Tabelle steht rechts; die Verbindungslinie symbolisiert die Relation.

Viele Software-Produkte gestatten das „Zeichnen“ von Tabellen in der angegebenen Form. Die sog. CASE-Tools (Computer Aided Software Engineering) fertigen sogar auf der Basis dieser Zeichnungen die oben erwähnten Skripte an, die dann - in SQLTalk - geladen und ausgeführt werden können, womit alle Tabellen, Views, Indizes und Benutzer auf Knopfdruck angelegt werden (dieses Verfahren wird von vielen, die zum erstenmal mit diesen Produkten in Berührung kommen, als umständlich empfunden; für kleinere Projekte mag das stimmen, bei umfangreichen Projekten ist es aber unabdingbar, damit zu arbeiten, so dass die Tätigkeiten reproduzierbar bleiben).

Tabelle oder Relation?

In vielen Lehrbüchern findet sich die Aussage „eine Tabelle ist eine Relation“. Das ist in dieser Form nicht richtig. Treffender ist die folgende Formulierung: „Eine Tabelle beschreibt eine Relation“ (Quelle: Informatik-Duden); dabei ist eine Relation eine Zuordnung von Elementen aus einer Teilmenge mit anderen Elementen aus einer anderen Teilmenge; beispielsweise eine Tabelle, in der die Firmennamen zusammen mit dem errechneten Jahresumsatz angezeigt werden.

Hinzu kommt, dass man **reale (echte) Tabellen** (wie die hier in der Datenbank ISLAND.DBS mitgelieferten Tabellen COMPANY, INVOICE, INVOICE_ITEM) trennt von anderen sog. **virtuellen Tabellen (oder Views);** ihre Namen sind in der Systemtabelle SYSTABLES gespeichert, sie können beispielsweise folgendermaßen angezeigt werden*:*

- die Namen der **realen Tabellen** können in SQLTalk mit der folgenden SQL-Anweisung angezeigt werden:

```
select * from systables where type = 'T';
```

- die Namen der **virtuellen Tabellen (oder Views**) lassen sich mit der folgenden Anweisung anzeigen:
  ```
  select * from systables where type = 'V';
  ```

Typischerweise wird im laufenden Betrieb einer Anwendung die Zahl der Tabellen ab einem bestimmten Zeitpunkt kaum noch zunehmen, dagegen werden durchaus noch neue Views angelegt, um neuartiger Fragestellungen zu beantworten oder um Daten zu analysieren.

Entität, Attribute, Relations

Tabellen unterscheiden sich in vielerlei Hinsicht: manche haben viele Spalten (oder Felder), manche haben viele Sätze (oder Zeilen). Die wichtigste Unterscheidung ist eine inhaltliche: In der einen Gruppe sind Daten in der Form enthalten, dass **ein ganzer Satz eine sogenannte Entität** beschreibt; dabei umfaßt eine Entität (engl. entity) Dinge der realen Welt, die bestimmte, beschreibbare Eigenschaften aufweisen, wie z. B. der Name eines Menschen oder die Nummer einer Rechnung. So hat beispielsweise ein Kunde die Merkmale Kundennummer, Firmenname, Nachname usw. Die Inhalte dieser Merkmale werden auch als **Attribute** bezeichnet. Beispielsweise werden von Mitarbeitern einer Firma die Attribute Personalnummer, Nachname, Gehalt usw. gespeichert. Wichtige Attribute eines Auftrages sind z. B. Auftragsnummer, Status des Auftrags (offen, bezahlt, gemahnt).

Hinzu kommen die Beziehungen (engl. relations, relationships) zwischen den einzelnen Attributen, so wie in der oben gezeigten Abbildung zwischen der Kundennummer links und der dazu passenden Datensätzen aus der Tabelle rechts.

SQL (Structured Query Language)

Auf die in Tabellen gespeicherten Daten wird - wie bereits oben erwähnt - mit der Sprache SQL zugegriffen. Sie bietet alle Kommandos die zum Einfügen (INSERT), zum Ändern (UPDATE), zum Löschen (DELETE) oder zum Abfragen (SELECT) nötig sind. Zu dieser Sprache folgt unten ein sehr ausführliches Kapitel (s. S. 111).

Index, Indizes

Häufig werden die Begriffe Index, Primärschlüssel und Fremd- oder Sekundärschlüssel durcheinandergeworfen. Während Primär- und Fremdschlüssel eine inhaltliche Bedeutung haben, ist ein Index etwas zum „Anfassen“:

- ein **Index** ist eine Hilfstabelle, die den Zugriff auf bestimmte, festzulegende Spalten einer Tabelle beschleunigt;
- ein **Primärschlüssel** wird beispielsweise auf der Kundennummer eines Kunden in einer Kunden-Parent-Tabellegelegt;
- von einem **Fremdschlüssel** (oder manchmal von **Sekundärschlüssel**) spricht man, wenn in einer Child-Tabelle das dort ebenfalls vorhandene Feld Kundennummer mit dem gleichnamigen Feld in der Parent-Tabelle verknüpft wird).

Damit ist auch klar, dass man einen Index in einer Tabelle mit nur 20 Sätzen nicht braucht. Auf der anderen Seite sollte man bei dem Entwurf von Tabellen und bei den späteren Abfragen herausfinden, über welche Felder üblicherweise auf die Daten zugegriffen wird, denn normalerweise erhalten diese einen Index. Damit beschleunigen Indizes folgendes:

- **Suchen** bzw. **Selektieren** von Datensätzen,

- **Sortieren** und **Gruppieren** von Datensätzen und das
- **Verknüpfen** von Daten.

Gute Kandidaten für einen Index sind z. B. Kunden-, Auftrags- oder Rechnungsnummern, einfach gesagt alle Felder, nach denen häufig ausgewertet, sortiert oder gesucht wird. Neben einem normalen Index kann man auch einen zusammengesetzten Index, bestehend aus zwei oder mehr Feldern oder Feldern und Funktionen anfertigen. Alles dies wird in dem Kapitel zu der Anweisung CREATE INDEX vorgestellt (s. S. 116).

Systemkatalog (oder auch Data Dictionary)

Jede Datenbank enthält einen SystemKatalog, das sind spezielle Tabellen, die Daten über andere Tabellen, Views usw. enthalten (eine Übersicht aller Systemtabellen findet sich im Anhang S. 291ff). Man verwendet diese Tabellen häufig, z. B. um herauszufinden

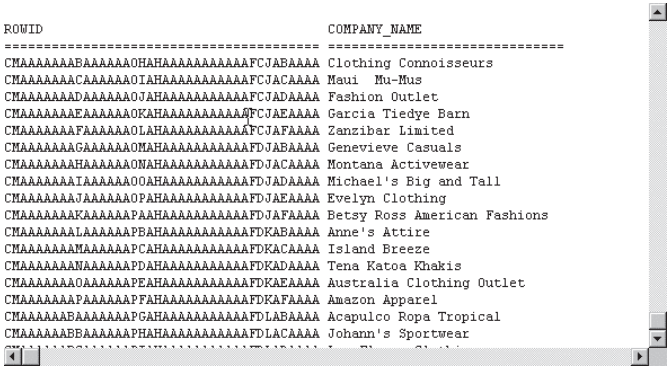
- welche Tabellen in der Datenbank enthalten sind (Tabelle SYSTABLES);
- welche Struktur (d. h. welche Felder, welche Felddatentypen) eine bestimmte Tabelle hat (Tabelle SYSCOLUMNS);
- welche Synonyme wurden für Tabellen festgelegt (Tabelle SYSSYNONYMES);
- welche Views vorhanden sind (Tabelle SYSVIEWS);
- welche Rechte hat ein bestimmter Anwender (z. B. Tabelle SYSCOLAUTH).

ROWID

Von großer Bedeutung bei der Arbeit mit SQLBase-Datenbanken ist die ROWID (zu deutsch etwa Satz-Identifikation). Zunächst ein Beispiel, wie eine solche ROWID überhaupt aussieht, hierzu die SQL-Anweisungen:

```
connect island;
select rowid, company_name
from company;
```

Die ROWID wird zusammen mit dem Inhalt des Feldes Company-Name aus der Tabelle Company angezeigt:

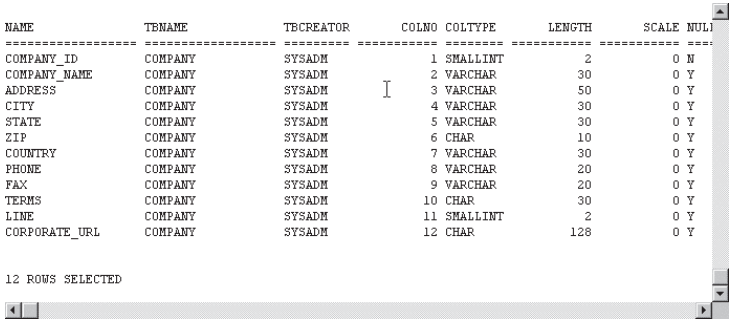


```
ROWID                                  COMPANY_NAME
====================================== ============================
CMAAAAAAABAAAAAAOHAHAAAAAAAAAAAAFCJABAAAA Clothing Connoisseurs
CMAAAAAAACAAAAAAOIAHAAAAAAAAAAAAFCJACAAAA Maui  Mu-Mus
CMAAAAAAADAAAAAAOJAHAAAAAAAAAAAAFCJADAAAA Fashion Outlet
CMAAAAAAAEAAAAAAOKAHAAAAAAAAAAAAFCJAEAAAA Garcia Tiedye Barn
CMAAAAAAAFAAAAAAOLAHAAAAAAAAAAAAFCJAFAAAA Zanzibar Limited
CMAAAAAAAGAAAAAAOMAHAAAAAAAAAAAAFDJABAAAA Genevieve Casuals
CMAAAAAAAHAAAAAAONAHAAAAAAAAAAAAFDJACAAAA Montana Activewear
CMAAAAAAATAAAAAAOOAHAAAAAAAAAAAAFDJADAAAA Michael's Big and Tall
CMAAAAAAAJAAAAAAOPAHAAAAAAAAAAAAFDJAEAAAA Evelyn Clothing
CMAAAAAAAKAAAAAAPAAHAAAAAAAAAAAAFDJAFAAAA Betsy Ross American Fashions
CMAAAAAAALAAAAAAPBAHAAAAAAAAAAAAFDKABAAAA Anne's Attire
CMAAAAAAAMAAAAAAPCAHAAAAAAAAAAAAFDKACAAAA Island Breeze
CMAAAAAAANAAAAAAPDAHAAAAAAAAAAAAFDKADAAAA Tena Katoa Khakis
CMAAAAAAAOAAAAAAPEAHAAAAAAAAAAAAFDKAEAAAA Australia Clothing Outlet
CMAAAAAAAPAAAAAAPFAHAAAAAAAAAAAAFDKAFAAAA Amazon Apparel
CMAAAAAABAAAAAAAPGAHAAAAAAAAAAAAFDLABAAAA Acapulco Ropa Tropical
CMAAAAAABBAAAAAAPHAHAAAAAAAAAAAAFDLACAAAA Johann's Sportwear
```

Auf den ersten Blick ist es verwunderlich, dass mit den Tabellendaten etwas angezeigt wird, was überhaupt nicht in der Tabelle steht, denn eine Spalte ROWID enthält die Tabelle Company nicht, wie der folgende Versuch mit der SQL-Anweisung

```
select * from syscolumns where tbname ='COMPANY';
```

zeigt; hier die Ergebnismenge:



NAME	TBNAME	TBCREATOR	COLNO	COLTYPE	LENGTH	SCALE	NULL
COMPANY_ID	COMPANY	SYSADM	1	SMALLINT	2	0	N
COMPANY_NAME	COMPANY	SYSADM	2	VARCHAR	30	0	Y
ADDRESS	COMPANY	SYSADM	3	VARCHAR	50	0	Y
CITY	COMPANY	SYSADM	4	VARCHAR	30	0	Y
STATE	COMPANY	SYSADM	5	VARCHAR	30	0	Y
ZIP	COMPANY	SYSADM	6	CHAR	10	0	Y
COUNTRY	COMPANY	SYSADM	7	VARCHAR	30	0	Y
PHONE	COMPANY	SYSADM	8	VARCHAR	20	0	Y
FAX	COMPANY	SYSADM	9	VARCHAR	20	0	Y
TERMS	COMPANY	SYSADM	10	CHAR	30	0	Y
LINE	COMPANY	SYSADM	11	SMALLINT	2	0	Y
CORPORATE_URL	COMPANY	SYSADM	12	CHAR	128	0	Y

12 ROWS SELECTED

Die ROWID wird von der SQLBase aus (Kontroll-) Daten zusammengestellt, die auf sog. pages enthalten sind. Jeder Anwender, der eine Select-Anweisung auf die Tabellendaten ausführen lässt, erhält eine Kopie dieser (zunächst einheitlichen) ROWID. Ändert nun ein Anwender einen Feldinhalt (mittels der UPDATE-Anweisung), dann wird die alte ROWID verworfen, SQLBase stellt eine neue zusammen. Ein anderer Anwender, der den gleichen Satz vor sich hat (der zwischenzeitlich durch den anderen Anwender geändert wurde) kann nun einfach seine (alte) ROWID mit der (neuen) ROWID vergleichen, die von der SQLBase gebildet wurde: weichen die beiden voneinander ab, dann weiß der Anwender (sofern der Programmierer entsprechend codiert hat), dass jemand anders den Satz geändert hat. Die Verwendung von ROWIDs spielt bei der Programmierung von Datenbank-Anwendungen mit der SQLBase eine große Rolle.

ROWID im Centura Team Developer

Codiert man eine Anwendung mit der SQLBase, verwendet man grundsätzlich die ROWID. Allerdings wird die ROWID nie angezeigt, wird aber in einer zusätzlichen Spalte eines Table-Windows oder in einem DataField gespeichert. Um die ROWID zusammen mit den Daten einer Tabelle zu lesen und sie in einem Table-Window anzuzeigen, geht man folgendermaßen vor:

- Spalten für die Daten einrichten und zusätzliche Spalte (unsichtbar) für die ROWID; als Name z. B. colROWID verwenden;
- Select-Anweisung ändern:
  ```
  select ..., ROWID from ... into ..., :colROWID ...
  ```

Bei UPDATE oder DELETE-Anweisungen wird jetzt nicht mehr auf die Gleichheit z. B. der Kundennummer geprüft, sondern es werden die ROWIDs miteinander verglichen:

```
update column set col1 = 'Test' where rowid = :colROWID
```

Falls ein Update möglich ist, wird ein X-Lock gesetzt und der Feldinhalt geändert, andernfalls kommt Fehlernummer 806, „invalid ROWID“. Diese Fehlernummer wird durch den Programmierer aufgefangen und die entsprechenden Massnahmen eingeleitet (Anwender mit Fenster informieren, im Fenster mehrere Optionen anbieten: abbrechen, Änderungen „durchdrücken“ usw.).

Cache und pages

Nahezu alle Computerprogramme verwenden für den schnelleren Zugriff auf Daten einen Speicher zum Puffern der Daten (cache), in dem eine Kopie der auf dem Datenträger befindlichen Daten (bzw. eines Teiles davon) abgelegt ist. Da eine Datenbank für mehrere User immer wieder Daten lesen muss, ist es nur wahrscheinlich, dass sich ein gewisser Teil der Daten bereits in diesem (schnellen RAM-) Speicher auf dem Server befindet und direkt aus dem Speicher zu dem Anwender geschickt werden kann. Demgegenüber ist der direkte Zugriff auf den Plattenspeicher, der durch Positionieren der Schreib-/Leseköpfe realisiert wird, wesentlich langsamer.

SQLBase prüft, nachdem ein Anwender eine Anweisung zu Lesen von Daten abgeschickt hat:

- befindet sich der Datensatz als Kopie schon im cache (d. h. wurde er schon gelesen) oder
- ist der Datensatz nur auf der Festplatte (nicht aber im cache).

Im ersten Fall wird der Datensatz aus dem cache gelesen oder im cache geändert, im zweiten Fall wird der Satz (bzw. die hierfür verwendete(n) page(s) vor der Bearbeitung gelesen und in den cache kopiert.

In Zusammenhang damit, dass Daten im Grunde in zwei Versionen vorliegen (die Daten auf der Festplatte einerseits und die Daten im cache andererseits - wenn man die Anwender und deren Hauptspeicher für einen Moment außen vor lässt) -, ist wichtig, dass das Datenbank-Management-System sogenannte *Checkpoints* hat (die zusätzlich zum *normalen cache-Handling* dafür sorgen, dass die geänderten Daten im cache auch zu Änderungen auf dem Datenträger führen). Checkpoints sind auf 1 Minute voreingestellte Zeitabstände, nach deren Ablauf geprüft wird, welche Daten geändert und demzufolge in die Datenbank geschrieben werden müssen (dieser Wert ist deswegen so klein eingestellt, damit im Falle eines Systemabsturzes möglichst wenig Daten verloren gehen). Man kann diesen Zeitabstand verändern, um das System schneller zu machen (s. hierzu die Anweisung SET CHECKPOINT in der Online-Hilfe von SQLTalk).

SYSDBSequence

SYSDBSequence ist ein Objekt, das in jeder Datenbank enthalten ist. In ihr befindet sich eine Zahl, die bei einer neuen Datenbank zunächst den Wert 0 enthält, dann aber hochgezählt wird. Man kann auf SYSDBSequence mit zwei „Funktionen“ zugreifen:

- SYSDBSequence.CurrVal liefert den aktuellen Inhalt,
- SYSDBSequence.NextVal liefert den nächsten freien verwendbaren Wert

Zwei kurze Beispiele hierzu:

Test mit einer neuen Datenbank

Um eine neue Datenbank anzulegen und zu öffnen, sollen folgende Anweisungen ausgeführt werden:

```
create database Test;
connect test;
```

Jetzt soll herausgefunden werden, womit SYSDBSequence gefüllt ist (hierzu muss in einer neuen – noch leeren – Datenbank beim ersten Aufruf „NextVal“ verwendet werden („CurrVal“ funktioniert nach dem ersten Aufruf von „NextVal“ auch):

```
select sysdbsequence.nextval
from systables;
```

Diese Anweisung liefert (für NextVal) den Wert 101, d. h. CurrVal ist derzeit mit 100 gefüllt. Bei jedem Ausführen der zuletzt genannten SQL-Anweisung wird der Wert von SYSDBSequence um 1 erhöht, alte - nicht verwendete - Werte stehen nicht mehr zur Verfügung.

Satz einfügen mit NextVal

Zurück in der Datenbank Island.DBS (ggf. „connect island;“ ausführen lassen) soll ein neuer Satz in die Tabelle Company eingefügt werden, ohne dass man sich selbst mit der Company_ID auseinandersetzen möchte. Man kann einfach den Inhalt von SYSDBSequence.NextVal verwenden:

```
insert into company
(company_id, company_name)
values
(SYSDBSequence.NextVal, 'Kniff & Co.');
```

Der neue Kunde erhält die ermittelte Company_ID von (hier:) 8007. Der einzige Wermutstropfen ist, dass NextVal und CurrCal nicht im Batch-Betrieb verwendet werden können, d. h.die folgende Anweisung, mit der zwei Datensätze eingefügt werden sollen, zu denen SQLBase selbst eine freie COMPANY_ID ermitteln sollte, lässt sich nicht ausführen:

```
insert into company
(Company_id, Company_name)
Values (:1, :2)
\
sysdbsequence.nextval , Keller
sysdbsequence.nextval , Goller
/
```

Sätze mit NextVal in Master-Detail-Tabellen einfügen

Durch die gleichzeitige Verwendung von CurrVal und NextVal kann man sogar Datensätze in Master-/Detail-Tabellen eingeben, indem man zunächst mit NextVal eine neue Nummer erzeugt, diese in die Master-Tabelle einfügt und danach mit CurrVal die gleiche Nummer verwendet, um in der Detail-Tabelle - unter Verwendung der gleichen Nummer - die zugehörigen Detail-Sätze eingibt.

Beispielsweise würde das für die hier selbst erstellten Tabellen Konten und Belege folgendermaßen aussehen:

```
insert into Konten
(KontoNr, Bezeichnung, Budget)
Values
( sysdbsequence.nextval, 'Kfz', 2000);

insert into belege
(belegnr, betreff, KontoNr)
Values
(77, 'Strafzettel', sysdbsequence.CurrVal);
```

Als KontoNr wird (hier) 8010 vom System vergeben, der danach eingefügte Detail-Satz erhält als Inhalt des Feldes „KontoNr“ die gleiche Zahl.