ESKER Tun® Plus

Tun SQL – Datenzugriff

ESKER®
Document Process Automation

Tun Plus 2009
Issued May 2008

Esker S.A., 10 rue des Émeraudes, 69006 Lyon, France
Tel: +33 (0)4.72.83.46.46 ♦ Fax: +33 (0)4.72.83.46.40 ♦ info@esker.fr ♦ www.esker.fr

Esker, Inc., 1212 Deming Way, Suite 350, Madison, WI 53717 USA
Tel: +1.608.828.6000 ♦ Fax: +1.608.828.6001 ♦ info@esker.com ♦ www.esker.com

Esker Australia Pty Ltd. (Lane Cove - NSW) ♦ Tel: +61 (0)2 8596 5100 ♦ info@esker.com.au ♦ www.esker.com.au
Esker GmbH (München) ♦ Tel: +49 (0) 89 700 887 0 ♦ info@esker.de ♦ www.esker.de
Esker Italia SRL (Milano) ♦ Tel: +39 02 57 77 39 1 ♦ info@esker.it ♦ www.esker.it
Esker Ibérica, S.L. (Madrid) ♦ Tel: +34 91 552 9265 ♦ info@esker.es ♦ www.esker.es
Esker UK Ltd. (Derby) ♦ Tel: +44 1332 54 8181 ♦ info@esker.co.uk ♦ www.esker.co.uk

VORWORT

Tun SQL - Datenzugriff ist eine Anwendungs- und Server Suite, die es PCs ermöglicht, im Client/Server Modus mit entfernten Datenbanken zu arbeiten (Informix, Oracle, Sybase, DB2, Progress und C-ISAM). **Tun SQL** benutzt die von Microsoft definierte ODBC Architektur.

Tun SQL läuft auf folgenden Plattformen: Windows 3.x, Windows 95, Windows 98, Windows NT 3.51, Windows NT 4.0, Windows 2000, Citrix WinFrame, Citrix MetaFrame und Windows NT TSE.

Tun SQL ist Bestandteil der Software-Produktreihe **Tun** (siehe nachfolgende Tabelle):

	Windows Version	**Komponenten einer Mehrbnutzer-Umgebung**
Esker TCP/IP Stack	TCP/IP-Kommunikations-Stacks für Windows 3.x (DLL)	N/A
Network Resource Access (Tun NET)	TCP/IP-Anwendungen (NIS, NFS-Client und -Server, PING, Druckerumleitung und gemeinsame Benutzung des Druckers, FTP-Client und -Server, TELNET, RSH-Client und -Server, TAR, WALL, TFTP, TIME)	TCP/IP-Anwendungen (NFS-Client und -Server, PING, Druckerumleitung und gemeinsame Benutzung des Druckers, FTP-Client und -Server, TELNET, RSH-Client, TAR, WALL)
Application Access (Tun EMUL)	Terminal-Emulator (asynchrone, IBM3270- und IBM5250-Emulation, 3287/3812-Drucker)	Terminal-Emulator (asynchrone, IBM3270- und IBM5250-Emulation, 3287/3812-Drucker)
Data Access (Tun SQL)	ODBC-Treiber für TCP/IP Client/Server-Modus (Oracle, Informix, Sybase, DB2, Progress und C-ISAM DBMS) und Datenbank-Revamping-Tool	ODBC-Treiber für TCP/IP Client/Server-Modus (Oracle, Informix, Sybase, DB2, Progress und C-ISAM DBMS) und Datenbank-Revamping-Tool
TCP/IP Network Services	NIS-Browser, Druckerumleitung und Druckersharing	Druckerumleitung und Druckersharing

Die meisten, in diesem Handbuch beschriebenen Funktionalitäten und Prozeduren gelten ebenfalls für Windows 3.x, Windows 95, Windows 98, Windows NT 3.51, Windows NT 4.0, Windows 2000 oder Citrix/Windows NT TSE. Einige Funktionalitäten und Prozeduren gelten jedoch nur für eine oder einige dieser Plattformen. In diesem Fall wird der betreffende Abschnitt wie folgt angegeben:

Windows 3.x

Windows 95 und Windows 98

Windows NT (Windows NT 3.51 und Windows NT 4.0 inklusive Mehrbenutzer-Umgebung, wenn nicht anders angegeben) und Windows 2000

Windows NT 4.0 inklusive Citrix/Windows NT TSE wenn nicht anders angegeben

Windows NT 3.51 inklusive Mehrbenutzer-Umgebung, wenn nicht anders angegeben

32-Bit Windows (Windows 95, Windows 98, Windows NT 3.51 und Windows NT 4.0 inklusive Mehrbenutzer-Umgebung, wenn nicht anders angegeben)

Mehrbenutzer-Umgebung

Mehrbenutzer-Umgebung ausgenommen

Tun SQL für Windows wird ebenfalls mit **Tun PLUS** geliefert, in dem alle o.g. Module enthalten sind. Bei der Installationsprozedur von **Tun PLUS** wird geraten, **Tun SQL** zu installieren.

Außer bei **Tun PLUS** für die Multi-User Windows -Version können Sie **Tun SQL** unabhängig von **Tun PLUS** installieren.

Hinweis
In diesem Handbuch sind die Funktionen in Windows 98 und identisch.

INHALTSVERZEICHNIS

TEIL 1
PRÄSENTATION UND NUTZUNG

1

EINFÜHRUNG IN TUN SQL

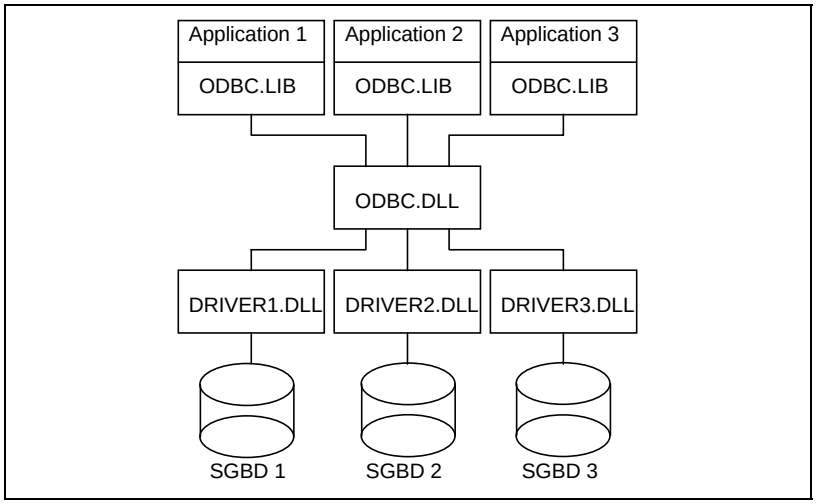
Der ODBC Mechanismus

In der Welt der Datenbanken benutzen Entwickler traditionell einen Mechanismusus, der "Embedded SQL" genannt wird, um eine Schnittstelle zwischen ihren Applikationen und einer spezifischen Datenbank zu haben. Der Embedded SQL Mechanismusus ermöglicht den Einschub von SQL Requests in C oder COBOL Programme. Er bietet den Vorteil, Applikationen für verschiedene Maschinen podertierbar zu machen.

Der Embedded SQL Mechanismusus hat jedoch auch einige Nachteile:

- Es gibt ebenso viele Embedded SQLs wie DBMS Engines auf dem Markt. Die SQL benutzenden Applikationen können jeweils nur mit einer DBMS gleichzeitig kommunizieren. Sie müssen neu geschrieben oder zumindest überarbeitet werden, wenn sie auf andere Datenbanken zugreifen sollen. Für Applikationen, die auf alle Datenbanken am Markt zugreifen sollen, gibt es keinen Weg mit dem Embedded SQL Mechanismus.
- Der Embedded SQL Mechanismus ist relativ unentwickelt, sehr restriktiv und schwer zu benutzen. Er läßt keine volle Nutzung aller Möglichkeiten einer Datenbank zu, und es ist teilweise sinnvoller das direkt mit dem DBMS gelieferte API zu nutzen..

Um diesen Nachteilen des Embedded SQL Mechanismus zu begegnen, hat Microsoft einen neuen Ansatz entwickelt, der auf dem ODBC Mechanismus (Open Database Connectivity) basiert, welcher seinerseits auf der WOSA Architektur (Windows Open System Architecture) aufsetzt.



ODBC ist ein vervollständigter Satz von C Funktionen, der es ermöglicht Daten in einem DBMS zu suchen oder zu aktualisieren. Diese Funktionen sind in einer DLL (Dynamic Link Bibliothek) gesammelt, die von allen WINDOWS Applikationen benutzt werden kann. Die Funktionen des ODBC DLL analysieren SQL Requests und lassen sie von ODBC Treibern bearbeiten, deren Aufgabe es ist, die Anfragen für die bestimmten APIs des zu benutzenden DBMS zu konvertieren. Ein ODBC Treiber gewährt Ihnen die Betrachtung der DBMS Schnittstelle und befähigt die Applikation sie wie jedes andere ODBC-kompatible DBMS zu benutzen.

Microsoft stellt die ODBC.DLL Bibliothek und alle zu deren Nutzung notwendigen Werkzeuge; sie liefern jedoch nicht die ODBC Treiber für alle DBMSs auf dem Markt. Microsoft gibt sich mit der Bereitstellung von Treibern ihrer proprietären Büroanwendungen zufrieden (Excel, Word, Access...). Spezifische ODBC Treiber für Datenbanken können direkt vom DBMS Anbieter oder von Drittherstellern, die sich auf dieses Gebiet spezialisieren, angeboten werden (Esker).

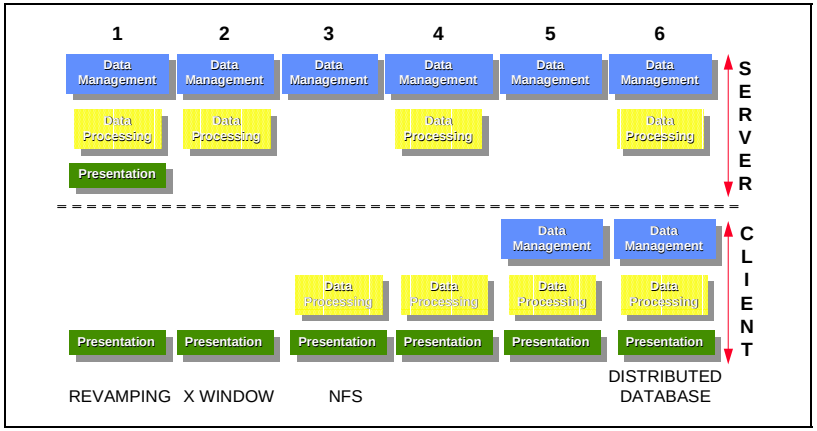
Schließlich bedeutet der ODBC Mechanismus maximale Interoperabilität. Eine einfache Windows Applikation kann auf verschiedene Management Systeme zugreifen, ohne notwendigerweise in dieser Absicht entwickelt worden zu sein. ODBC erlaubt dem Entwickler seine Anwendungen ohne Rücksicht auf das letztlich genutzte DBMS zu programmmieren, zu kompilieren und zu liefern.
Die Anwender müssen lediglich den richtigen Treiber besitzen, so daß die Applikation, mit der sie arbeiten, mit der Datenbank ihrer Wahl kooperieren kann.

ODBC ist ein wertvoller Mechanismus für Multi-Domain Applikationen wie Tabellenkalkulations-, Textverarbeitungs-applikationen und Entwicklungs-werkzeuge, die Informationen von jeglichem DBMS manipulieren können, ohne a prior*i* zu wissen, welche DBMS tatsächlich genutzt wird.

Das Client/Server Modell

Seit einigen Jahren ist Client/Server ein Begriff, den jeder in der Computerindustrie im Munde führt. Im weitesten Sinne stellt das Client/Server Modell ein Konzept dar, in dem zumindest zwei Einheiten zur Bereitstellung eines bestimmten Dienstes herangezogen werden.

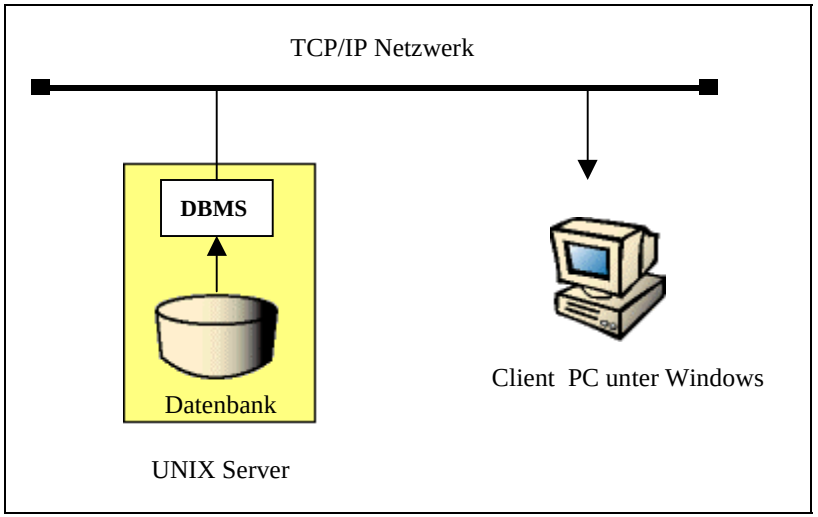
Die "Gartner Group" unterscheidet sechs grundsätzliche Typen der Client/Server Modus Applikationen, gemäß der Anzahl von Funktionen die vom Client auf dem Server ausgeführt werden.



Die Applikationen, die die geringsten Anforferungen an den "Client" stellen, sind "Revamping " Applikationen oder X-Window Server. Die Applikationen, mit den höchsten Anforderungen an den "Client" sind solche, die verteilte Datenbanken nutzen.

Wenn in der Computerindustrie vom Client/Server Modell die Rede ist, sind sehr oft Applikationen, die verteilte Datenbanken betreiben, gemeint.

Die allgemeinste Darstellung dieses Modus ist im folgenden Schema zu sehen:



Ein "Client" PC hat ein traditionelles Grafikmodus Managementsystem. Die Daten werden zentral auf einem UNIX Server gehalten und durch ein relationales Datenbank Management System (RDBMS) verwaltet. Um Daten abzufragen oder zu aktualisieren sendet der Client einen SQL Request, der es ausführt, um die Antwodert über das Netzwerk an den Client zurückzuschicken.

Die grundsätzlichen Vorteile dieser Architektur sind folgende:

- Der Endanwender hat eine grafische benutzerfreundliche Mensch/Maschine Schnittstelle auf seinem PC, die das Windows Betriebssystem nutzt.

- Der PC kann für andere Dinge als die normalen Applikationen (Büroautomation, Kalkulationen, persönliche Applikationen...) genutzt werden.
- Obwohl er seine eigene Maschine hat, kann der Benutzer gleichzeitig mit anderen auf zentralisierte Daten auf Servern zugreifen.
- Der Server spart sich den Applikationsteil des Rechnens und kann all seine Leistung auf die Bereitstellung von Daten konzentrieren. Es herrscht ein gesundes Gleichgewicht zwischen den Arbeitslasten des Clients und des Servers.
- Das Netzwerk wird nur mit dem Versand wesentlicher Daten belastet; es wird nicht unnötig für den Transport von Darstellungsdaten mißbraucht.

Der SQL Client/Server Modus wie oben beschrieben, ist eine modernisierte Form des Transaktionsmodus, der immer noch in Umgebungen mit Mainframes und synchronen Terminals genutzt wird.

ODBC und das SQL Client/Server Modell

Insoweit als der ODBC Mechanismus Zugriff auf alle Datenmanagement Systeme erlaubt, kann er auch auf Remote Datenbanken genutzt werden. In diesem Zusammenhang gestattet ODBC einer Windows Applikation DBMS jeglichen Ursprungs zu nutzen. Es erlaubt multi-domain Applikationen wie Excel, Word oder Access die zentralisierten Daten des Unternehmens zu nutzen. Im Client/Server Umfeld wird die Anzahl der Applikationen, die auf Unternehmensdaten zugreifen können, vervielfacht.

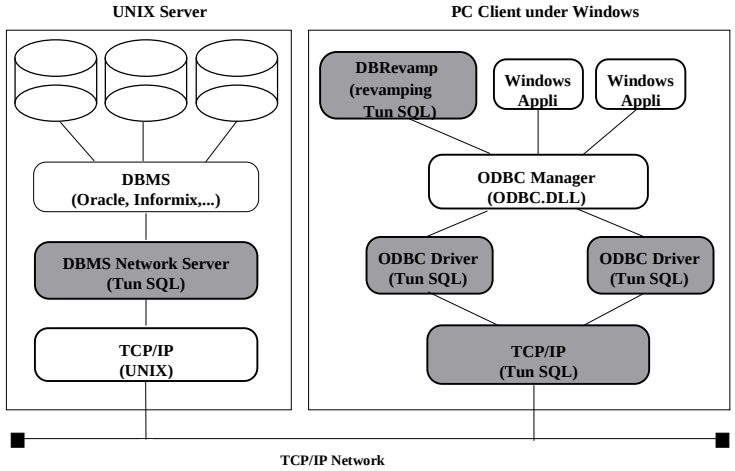
In jedem Fall ist die Implementation einer solchen Architektur heute kein Kinderspiel. Ein Benutzer, der bereits über einen lokalen PC, einen UNIX Server und ein DBMS verfügt, muß noch die folgenden Komponenten hinzufügen, um eine SQL Client/Server Architektur mit ODBC funktionsfähig zu machen:

- Den Netzwerk Server Teil des DBMS (Informix Net, Sql Net...).
- Den PC Client Teil des DBMS (Informix Net PC, Sql Net PC...).
- Den entsprechenden ODBC Treiber.
- Winsock-kompatible TCP/IP Stacks für den PC.

DBMS Anbieter liefern immer die ersten beiden Komponenten. Das gilt nicht immer für den ODBC Treiber und niemals für den TCP/IP Stack. Man ist daher dazu gezwungen die letzten beiden Komponenten woanders zu erwerben, wobei speziell das Problem ihrer Kompatibiltät ins Gewicht fällt.

Tun SQL

Tun SQL wurde entwickelt, um dieses Problem umfassend zu lösen. Es enthält in einem einzigen homogenen Softwarepaket alle oben erwähnten Komponenten sowie leistungsstarkes Datenbank-Revamping und einen virtuellen ODBC Treiber zum Zugriff auf die revampten Datenbanken. Sogar die Netzwerkkomponenten des DBMS (Client und Server) sind in **Tun SQL** enthalten. Dies repräsentiert auch ein enorme Ersparnis bei der Ausstattung vieler PCs. Es vereinfacht besonders die Implementation von Client/Server Architektur sehr.



➢ Nur ein ODBC Treiber für die meisten DBMS auf dem Markt

Um die für einen Client PC notwendigen Softwareprodukte zu begrenzen, enthält **Tun SQL** einen einzigen ODBC Treiber um auf die folgenden DBMS unterschiedslos zuzugreifen:

- Oracle Version 7.
- Informix Version 5 und Version 7.
- Sybase Version 10.
- DB2 Version 2.
- Progress Version 6, Version 7 und Version 8.
- C-ISAM Versionen 4 bis 7.

➢ Datenbank-Revamping

Tun SQL kann mit Hilfe einer integrierten anwenderfreundlichen Applikation die Tabellen in einer Datenbank umdefinieren und sie Endbenutzern zugänglicher machen (durch angepaßte Datenbankreorganisation, Änderung von Tabellen- und Feldnamen und vorbesetzte Funktionen).

➢ Ein virtueller ODBC Treiber für revampte Datenbanken

Um durch Revamping umdefinierte Datenbanken zu benutzen, enthält **Tun SQL** einen virtuellen ODBC Treiber, der Anfragen an virtuelle Tabellen in Anfragen, die ein normaler ODBC Treiber abhandeln kann, zu übersetzen.

➢ Der Server Part des DBMS ist in Tun SQL enthalten

Der Server Part jedes DBMS wird unter UNIX installiert und wird standardmäßig mit **Tun SQL** für die folgenden Betriebssysteme geliefert:

- ScoUnix 3.2x v.4.2 and 5.0
- SunOs 4.1.3
- Solaris 2.5
- AIX 3.2 and 4.1
- HP-UX 9.x and 10.x
- OSF1 v.3.2

Dieses Feature erspart dem **Tun SQL** Benutzer die Kosten für den Server Teil des benutzten DBMS.

➢ TCP/IP Stacks standardmäßig enthalten (16-Bit Windows)


Win 3.x

Wie alle Software der **Tun** Reihe wird **Tun SQL** standardmäßig mit **Esker TCP/IP Stack** geliefert. Der Stack hat ausgezeichnete Performancewerte und ist mit allen **Tun SQL** Komponenten getestet. Die Einbindung des Stacks in das **Tun SQL** Paket erspart dem Benutzer die anderweitige Anschaffung solcher Komponenten.

➢ Einfachhheit der Installation und Administration

Der Zweck von **Tun SQL** ist die Bereitstellung der Implementation von Client/Server Architektur auf der Basis von Windows und Unix. Konsequenterweise hat **Tun SQL** eine einfache Installationsprozedur für Unix und Windows und eine vollständige Dokumentation.

Zusätzlich zu den ODBC Treibern werden drei Windows Applikationen zum Test und zur Implementation des Client/Server geliefert :

- **Tun DB Show** wird zum Client/Server Verbindungstest benutzt. Die Applikation kann auf einem Unix Server herausfinden welche DBMS installiert sind und sie auf die enthaltenen Datenbanken hin durchsuchen.
- **Tun DB Script** kann SQL Batchdateien unter Windows ausführen, um auf entfernten DBMS Datenbanken zu erzeugen.
- **Tun DB Map** zum Anlegen oder Ändern von Zeichenübersetzungstabellen (character conversion tables).

In Ergänzung zu der Sicherheit, die durch Unix und die verschiedenen DBMS ohnehin geboten wird verfügt **Tun SQL** über einen Mechanismus, der einigen Windows Applikationen den Zugriff auf bestimmte sensible Datenbanken verwähren kann.

Schließlich erlaubt die Integration von NIS (Network Information Service) in **Tun SQL** die zentrale Verwaltung von Netzwerk-Ressourcen und erleichtert den Zugriff auf die remoten Ressourcen. Nähere Informationen zu NIS entnehmen Sie bitte dem Benutzerhandbuch der **TCP/IP Network Services**.

➢ Tun SQL Treiberkonformität

Der **Tun SQL** Treiber unterstützt alle Level 1 Funktionen und einige der Level 2 Funktionen. Die "Microsoft ODBC Cursor Library" wir dinkl. Treiber mitgeliefert. Obwohl diese Library nur statische und "forward only" Cursor unterstützt, ist dies für die meisten Applikationen ausreichend.

KONFIGURATION UND NUTZUNG UNTER WINDOWS

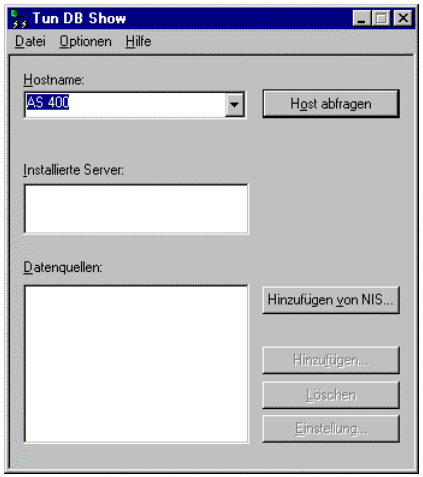
Prüfung der korrekten Funktionsweise von Tun SQL

➢ Tun DB Show ausführen

Nach Installation und Konfiguration von **Tun SQL** unter Windows und unter UNIX, muß die korrekte Funktionsweise überprüft werden. Die kann mittels der Applikation **Tun DB Show** geschehen.

Starten Sie das Programm durch Klicken auf das **Tun DB Show** Symbol in der **Data Access** Gruppe (**Start** Menü, **Programme**, **Esker Tun** in Windows 95/98/2000 und Windows NT).

Daraufhin erscheint folgendes Fenster:

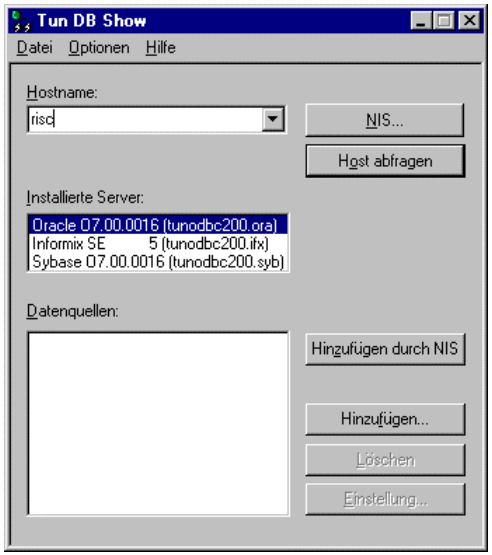


Dieses Hilfsprogramm kann benutzt werden, um einen UNIX Host im Netzwerk auf die Existenz eines oder mehrerer **Tun SQL** UNIX Servers zu prüfen. Geben Sie den Namen oder die IP-Adresse des Hosts im Feld **Hostname** ein, oder wählen Sie den Host aus der Auswahlliste (Diese Liste zeigt die Namen der Server aus der **hosts**-Datei und vom NIS Server.)

Wenn die Applikation **Tun NIS** auf dem PC installiert ist und der Netzwerkadministrator die NIS Tabellen konfiguriert hat, kann der **NIS** Button zum Zugriff auf die im Netzwerk installierten Server benutzt werden. Informationen zur Konfiguration von **Tun NIS** finden Sie im Handbuch **TCP/IP Network Services** oder **Tun NET**.

Drücken sie dann den **Host abfragen** Button.

Wenn einer oder mehrere **Tun SQL** UNIX Server auf der remoten Maschine korrekt installiert sind, sollte die Liste (mindestens eine Zeile) **Installierte Server** wie folgt erscheinen:



Jede Zeile enthält die folgende Information:

- Der Name des DBMS mit dem der **Tun SQL** UNIX Server kommuniziert.
- Die Versionsnummer des DBMS.

- Der Name der ausführbaren UNIX Datei, die als Server funktioniert (**tunodbc200.ora**, zum Beispiel).

Hinweis:
Mit einigen DBMS (Informix On-Line, zum Beispiel) wird bei der Anwahl von Server in der Liste die Liste von Datenbanken, die von einem DBMS verwaltet werden, in der entsprechenden Spalte (Datenbanken) angezeigt.

Wenn sich keine **Tun SQL** Server in der Liste befinden, bedeutet das, daß es bei der Installation Probleme gegeben hat. In diesem Fall müssen alle in den vorherigen Kapiteln beschriebenen Vorgänge und Tests erneut ausgeführt werden.

➢ Parameter

Im **Optionen** Menü in **Tun DB Show** können Sie:

- die Einstellungen der Dienste verändern.
- die Benutzung eines Proxy Server angeben.

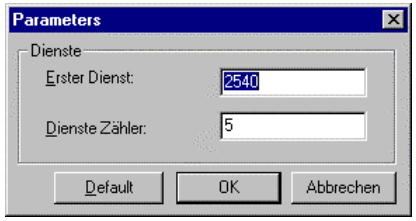
Einstellungen der Dienste

Eine Servicenummer ab 5370 ist jedem **Tun SQL** Server Prozeß zugeordnet. Damit **Tun SQL** einwandfrei funktioniert, müssen Sie den ersten Dienst und die Anzahl weiterer möglicher Dienste angeben. Dies ermöglicht **Tun SQL** mit den verschiedenen Datenbanksystemen über den **Tun SQL** Server zu kommunizieren. Die Voreinstellungen sind 5370 für den **Ersten Dienst** und 5 für den **Dienste Zähler**.

Die Liste der Services sieht folgendermaßen aus:

- Oracle 5370
- Informix 5371
- Sybase 5372
- DB2/RS6000 5373
- Progress 6 5374
- Progress 7 5375
- C-ISAM 5376
- DB2 für MVS 5377
- Progress 8 5378

Wählen Sie **Optionen➔Parameters...** aus dem Hauptmenü. Die folgende Dialogbox erscheint:

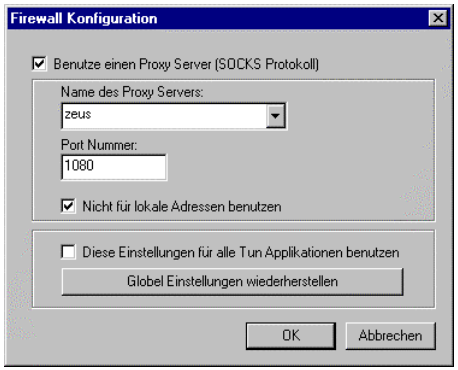


Benutzung eines Proxy Servers / Firewall

Wenn Sie einen Proxy Server in **Tun DBRevamp** definieren, wird jeder Zugriff auf den Datenbankserver über den Proxy Server geleitet.

Um die Firewall Parameter zu ändern (IP-Addresse, Portnummer, etc.), wählen Sie **Optionen➔Firewall** aus dem Hauptmenü.

Es erscheint die folgende Dialogbox:



Markieren Sie dir **Benutze einen Proxy Server** Checkbox.

Geben Sie den Namen oder die IP-Adresse der Servers an. Wenn Sie mit DNS arbeiten brauchen Sie nur den Namen angeben. Sie können ebenfalls einen Namen aus dem Drop-Down Menü auswählen, indem Sie auf den Pfeil klicken. Diese Liste enthält Einträge von Servern, die in entweder Ihrer HOSTS Datei oder in NIS (falls verwendet) definiert sind (NIS Ressourcen sind gelb dargestellt).

Geben Sie bitte ebenfalls die SOCKS Portnummer an (normalerweise 1080).

Um den Umweg über den Firewall bei lokalen Verbindungen zu vermeiden können Sie **Nicht für lokale Adressen benutzen** wählen.

Diese Firewall Einstellungen können Sie für alle **Tun** Applikationen verbindlich machen, indem Sie **Diese Einstellungen für alle Tun Applikationen benutzen** markieren. Um die allgemeinen Einstellungen für alle Tun Programme wieder herzustellen (z.B. nach einer speziellen Einstellung in **Tun NFS**) klicken Sie auf **Globale Einstellungen wiederherstellen**.

Erzeugung einer Datenbank

Damit die Anwender sich mit der ODBC Funktionsweise vertraut machen können, enthält **Tun SQL** einige praktische Beispiele. Um die mit dem **Tun SQL** Softwarepaket gelieferten Beispiele zu nutzen, muß eine spezielle Datenbank in einem der verfügbaren DBMSs erzeugt werden. Dies kann mit Hilfe der betreffenden Datenbank Werkzeuge geschehen.

Wenn unter Umständen die Erzeugung einer neuen Datenbank in einem DBMS (Oracle) sehr schwierig ist, kann auch eine existierende Datenbank benutzt werden, soweit sie keine sensiblen Daten enthält. Es empfiehlt sich die neu erzeugte Datenbank **tunsqldemo** zu nennen, um den nächsten Teil der Dokumentation besser verstehen zu können.

Erzeugung einer Datenquelle

➢ Einführung in Datenquellen

Damit dieser oder jener Treiber und diese oder jene Datenbank benutzt werden können, müssen ODBC-fähige Applikationen eine Datenquelle erkennen.

Datenquelle ist ein Schlüsselbegriff, der den Namen des benutzten ODBC Treibers (zum Beispiel, **tunodb32.dll**) und die zu seiner Funktionsfähigkeit notwendige Information enthält.

Diese Information ist folgende:

- Der Name oder die IP Adresse des UNIX Host.
- Der DBMS-Typ (Oracle, Informix, Sybase, DB2, Progress, C-ISAM).
- Der Name der Datenbank.
- Kommentare.
- Zusatzinformation.

➢ Erzeugung einer Datenquelle

Die mit **Tun SQL** gelieferten Beispiele verwenden die gleiche Datenquelle (ausgenommen die Beispiele für Datenbank-Revamping). Sie müssen diese Datenquelle erzeugen (Siehe "**Erste Schritte mit Tun**"), bevor Sie die Beispiele nutzen können.

Für die Erzeugung einer Datenquelle stehen zur Verfügung:

- **Tun DB Show**.
- **ODBC DatenQuellen-Administrator** (Windows Utility).

Erzeugen einer Datenquelle mit Tun DB Show

Starten Sie erneut **Tun DB Show** und gehen Sie wie folgt vor:

- Geben Sie den Namen oder die IP-Adresse des Servers ein, auf dem die Datenbank installiert ist und für die Sie eine Datenquelle erzeugen möchten.
- Klicken Sie **Host abfragen**, um die auf dem UNIX Host installierten SQL Server anzuzeigen.
- Wählen Sie den DBMS-Server mit der Datenbank, für die Sie eine Datenquelle erzeugen möchten.
- Klicken Sie **Hinzufügen...**.

Wenn **Tun NIS** auf dem PC installiert ist und der Netzwerkadministrator die NIS Tabellen konfiguriert hat, können Sie den Button **Hinzufügen aus NIS...** verwenden, um über das Netzwerk auf die Datenquelle zuzugreifen. Informationen zur Konfiguration von **Tun NIS** entnehmen Sie bitte dem Handbuch **TCP/IP Network Services** oder **Tun NET**.

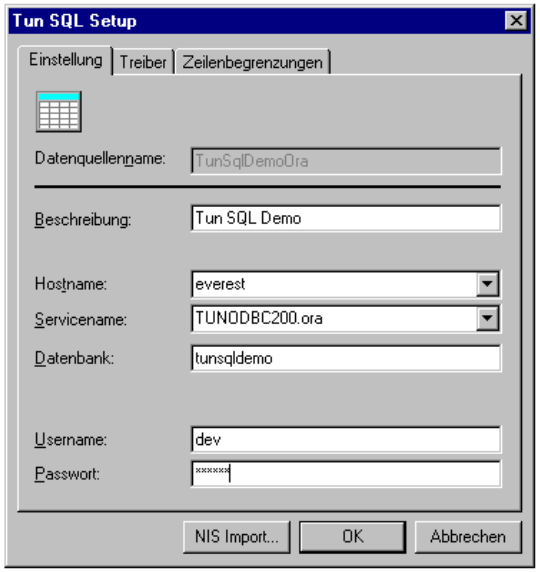
Erzeugung einer Datenquelle mit dem ODBC Administrator

Öffnen Sie die **Systemsteuerung** und klicken Sie das ODBC Symbol (32 Bit ODBC in Windows 32 Bit). Klicken Sie den Button **Hinzufügen** in der angezeigten Dialogbox.

Wählen Sie den ODBC Treiber **Tun32 Driver**.

Konfigurieren der Datenqelle

Klicken Sie den Button **Hinzufügen** in **Tun DB Show** oder im **ODBC Administrator**, um die folgende Dialogbox anzeigen zu lassen:



➢ Einstellung

Die einzelnen Felder dieser Box haben folgende Bedeutung:

Datenquellenname

Das Symbol  stellt das Feld einschließlich des Namens der Datenquelle, wie er durch die ODBC-fähigen Anwendungen benutzt wird,dar. Um auf die Beispieldatenbank zugreifen zu können, muß die Datenquelle vorhanden sein.

TunSqlDemoIfx	Für Informix On-Line
TunSqlDemoIse	Für Informix SE
TunSqlDemoOra	Für Oracle
TunSqlDemoSyb	Für Sybase
TunSqlDemoDB2	Für DB2
TunSqlDemoPro	Für Progress

Die anderen Services sind wie folgt festgelegt (werden im Beispiel nicht benutzt):

```
tunodbc200.pro7      5375/tcp      # Tun-SQL PROGRESS7
tunodbc200.ism       5376/tcp      # Tun-SQL C-ISAM
tunodbc200.mvs       5377/tcp      # Tun-SQL DB2/MVS
tunodbc200.pro8      5378/tcp      # Tun-SQL PROGRESS8
```

Beschreibung

Dieses Feld enthält einen mit der Datenquelle verbundenen Kommentar.

Hostname

Dieser Name enthält die IP Adresse oder den Namen des Hosts, auf dem die Datenbank, auf die der Nutzer zugreifen will, installiert ist.

Servicename

Dieses Feld enthält den Namen des **Tun SQL** Server Prozesses, der mit dem DBMS, indem die Datenbank, die der Anwender benutzen will, erzeugt wurde, verbunden ist (zum Beispiel, tunodbc200.ora).

Wenn Sie andere TCPIP Stacks als **TCP/IP Stack** benutzen, sollten Sie die Dienste (Datei services) in der verwendeten TCP/IP Software mit folgenden Werten vervollständigen:

```
tunodbc200.ora      5370/tcp     # Tun-SQL ORACLE
tunodbc200.ifx      5371/tcp     # Tun-SQL INFORMIX
tunodbc200.syb      5372/tcp     # Tun-SQL SYBASE
tunodbc200.db2      5373/tcp     # Tun-SQL DB2
tunodbc200.pro      5374/tcp     # Tun-SQL PROGRESS
```

Die anderen Services sind wie folgt festgelegt (werden im Beispiel nicht benutzt):

```
tunodbc200.pro7     5375/tcp    # Tun-SQL PROGRESS7
tunodbc200.ism      5376/tcp    # Tun-SQL C-ISAM
tunodbc200.mvs      5377/tcp    # Tun-SQL DB2/MVS
tunodbc200.pro8     5378/tcp    # Tun-SQL PROGRESS8
```

Datenbank

Geben Sie den Namen der Datenbank ein, die Sie benutzen möchten. Für die Beispieldatenbank geben Sie bitte **tunsqldemo** ein (wird mit **Tun SQL** geliefert).

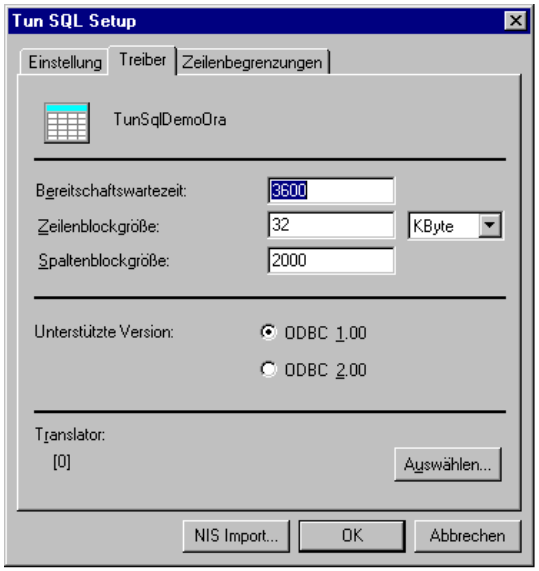
Username

Dieses Feld enthält den Namen eines Benutzers, der zum Zugriff auf die Datenbank berechtigt ist.

Passwort

Geben Sie das dem Benutzer zugewiesene Passwort ein.

Klicken Sie auf die **Treiber** Tabelle, um das ODBC Treiber Konfigurationsfenster anzuzeigen:



Bereitschaftswartezeit

Da der PC eine Maschine ist, die häufigen Hard- und Softwareausfällen unterliegt, muß der **Tun SQL** UNIX Server regelmäßig prüfen, ob der PC betriebsbereit ist. Zu diesem Zweck werden regelmäßig Pakete zum PC gesendet. Wenn er nicht innerhalb von **n** Sekunden antwortet, wird der Prozeß abgebrochen. Dieses Feld bestimmt die max. Antwortzeit (der Vorgabewert ist eine Stunde).

Zeilenblockgröße

Zeigt die Größe der während einer SQL "select" Operation aus einer Tabelle extrahierten Datenpakete an. Der Wert kann in Kilobytes oder Zeilenanzahl angegeben werden.

Wenn der Wert 1 ist, wird ein TCP Paket pro Zeile durchsucht. Entspricht der Wert 100, werden die Zeilen bis zur Anzahl der tatsächlich zu durchsuchenden Zeilen in 100er Paketen zusammengefaßt. Mit diesem Wert kann der Netzwerkverkehr optimiert werden. Der optimale Wert liegt zwischen 50 und 150. Der Standardwert liegt bei 32 KB.

Spaltenblockgröße

Zeigt die Fragmentierungseinheit für den Empfang sehr breiter Spalten aus der Datenbank (große Mengen an Text oder Grafiken) an. Wenn dieser Wert zu niedrig angesetzt wird, erhöht sich der Netzwerkverkehr signifikant.

Unterstützte Version

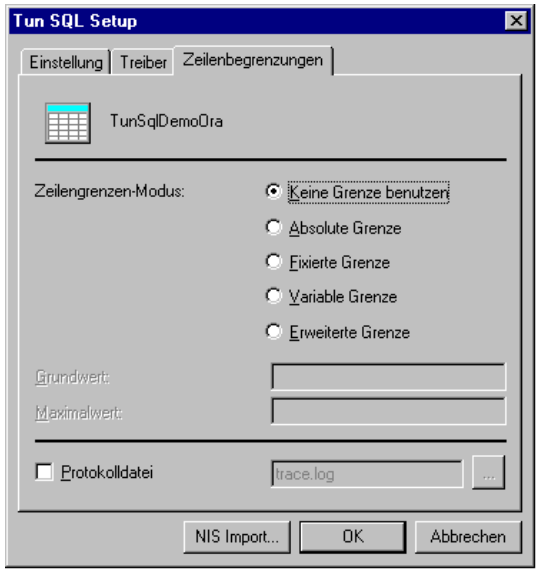
Zur Zeit gibt es zwei Versionen der ODBC API, mit den Versionsständen 1.00 und 2.00. Einige Applikationen sind nur mit ODBC Version 1.00 (Microsofts "Access") kompatibel und arbeiten nicht mit Treibern einer höheren Version. Der **Tun SQL** ODBC Treiber, der mit Version 2.00 kompatibel ist, kann auch Version 1.00 emulieren, so daß diese Applikationen trotzdem darauf laufen können. Wählen Sie in der Checkbox die den Anforderungen Ihrer Applikation genügende Treibergeneration.

Translator

Ausgehend von der unterschiedlichen Notation, die beispielsweise Umlaute in der Windows (CP850) und UNIX (ISO8859) Umgebung haben, ist es teilweise vonnöten Zeichkonvertierungstabellen für die ODBC Treiber einzustellen. Der **Auswählen...** Button dient dem Anwender zur Auswahl der gewünschten Konvertierungstabelle. In der Beispieldatenbank kann dieses Feld ignoriert werden, da alle enthaltenen Texten englisch sind.

Hinweis:
Die Konvertierungstabellen können mit der Applikation **Tun DB Map** editiert oder erzeugt werden.

Klicken Sie auf die **Zeilenbegrenzungen** Tabelle, um diese Dialogbox anzuzeigen:



Zeilengrenzen-Modus

Einige Büroautomations-Applikationen erlauben dem Anwender die Erstellung eigener SQL Requests. In anderen Fällen neigen bestimmte Applikationen dazu den gesamten Inhalt einer Tabelle zu lesen, bevor sie die Daten am Bildschirm darstellen. Dies ist kein Problem, solange es sich um kleine lokale Datenmengen handelt. Sind jedoch riesige Tabellen einer zentralen Datenbank betroffen, kann das zu unüberwindbaren Problemen führen. In diesem Fall wächst der Netzwerkverkehr enorm an und der Speicher des Pcs läuft über. Der PC muß häufig neu gestartet werden nach derartigen "select" Requests.

Um dieses Problem zu kompensieren, enthält der **Tun SQL** ODBC Treiber die Begrenzungsoptionen, die über die **Zeilengrenzen** Tabelle bestimmt werden können.

Fünf Typen von Grenzen werden vom **Tun SQL** ODBC Treiber erkannt:

Keine Grenze benutzen	Keine Grenze wird durch den ODBC Treiber gesetzt
Absolute Grenze	Der ODBC Treiber wird nicht mehr als **n** Zeilen während eines **select** Requests laden. Der Anwender erhält keine Nachricht.
Fixierte Grenze	Der ODBC Treiber wird nicht mehr als **n** Zeilen während eines **select** Requests laden. Der Anwender erhält eine Bildschirmmeldung.
Variable Grenze	Der ODBC Treiber wird nicht mehr als **n** Zeilen während eines **select** Requests laden. Der Anwender erhält eine Bildschirmmeldung, die ihm das Laden von mehr innerhalb der Grenzen des **Maximalwertes** nahelegt.
Erweiterte Grenze	Der ODBC Treiber wird nicht mehr als **n** Zeilen während eines **select** Requests laden. Der Anwender erhält eine Meldung, die ihm das Laden von mehr ohne einen Maximalwert anzeigt. In diesem Fall ist die Meldung lediglich eine Warnung.

Protokolldatei

Sie können die Checkbox **Protokolldatei** wählen, um die Verfolgung Ihrer SQL Anfragen in eine später zu untersuchende "**.log**" Datei zu sichern. Diese Datei zeigt, was der ODBC Treiber beim Erhalt einer SQL Anfrage tut.

Klicken Sie den Button **...**, um das Verzeichnis, in dem diese Datei gesichert werden soll, zu bestimmen.

NIS

Wenn **Tun NIS** auf dem PC installiert ist und der Netzwerkadministrator die NIS Tabellen konfiguriert hat, können Sie auf den Button **Import NIS...** klicken, um über das Netzwerk auf die Datenquelle zuzugreifen. Informationen zur Konfiguration von Tun NIS (falls Sie Tun NET nicht verwenden) entnehmen Sie bitte dem Handbuch **TCP/IP Network Services** oder dem Kapitel "**Der NIS Browser**" im **Tun NET** Handbuch.

Als NIS Administrator können Sie diejenigen Datenquellen exportieren, die Sie für den NIS Server konfiguriert haben. Klicken Sie den Button **Export NIS...**.

Hinweise:

- In diesem Kapitel wurden Sie dazu aufgefordert eine Datenquelle namens **tunsqldemoXXX**, auf die alle mit dem **Tun SQL** Software Paket gelieferten Beispiele sich beziehen, anzulegen.
- Wenn Sie **Tun SQL** mit anderen Applikationen und Datenbanken nutzen wollen, muß jeweils die entsprechende Datenquelle erzeugt werden. Als allgemeine Regel gilt, daß eine bestimmte Datenquelle für jede Applikation und für jede Datenbank erzeugt werden muß.
- Eine Datenquelle kann direkt durch Nutzung der ODBC Applikation im **Konfigurationspanel** erzeugt werden.

Übertragung der Demo-Datenbank

Das **Tun SQL** Paket wird mit einer Beispieldatenbank zur Nutzung mit den beigefügten Beispielen geliefert. Diese Datenbank muß vom PC auf die Datenbank **tunsqldemo**, die Sie in den vorstehenden Abschnitten erzeugen sollten, heruntergeladen werden.

Hinweis:
Die Zeichen XXX im Datenquellennamen stehen für einen der folgenden Werte:

- **Ifx** For Informix On-Line
- **Ise** For Informix SE
- **Ora** For Oracle
- **Syb** For Sybase

Um dieses Herunterladen durchzuführen, starten Sie das Programm durch Klicken auf das **Tun DB Script** Symbol in der **Data Access** Gruppe (**Start** Menü, **Programme**, **Esker Tun** in Windows 95/98/2000 und Windows NT).

Beim Start dieser Applikation erscheint folgendes Fenster:

➢ Laden der SQL Batchdatei zur Erzeugung der Datenbank

Die Datenbank Ihrer Wahl kann durch die Option **Datei→Öffnen** oder Anklicken des Button  geladen werden. Um die Beispieldatenbank herunterzuladen, muß die Datei **\Demo\Db\XXXcreate.sql** aus dem Installationsverzeichnis von **Tun SQL** geladen werden.

➢ Verbindung mit der Datenquelle

Bevor Sie die SQL Batchdatei laden können, müssen Sie eine Verbindung mit einer Datenquelle aufbauen. Um das zu tun, klicken Sie auf den Button  oder benutzen Sie die Option **Datenbank→Verbinden...**. Diese Aktion zeigt eine Dialogbox an, die nach dem Namen der Datenquelle fragt. Ist die Verbindung aufgebaut, wird die Batchdatei laufen.

Um die Beispieldatenbank herunterzuladen, selektieren Sie die zuvor definierte Datenquelle **TunSqlDemoXXX.**

➢ Ausführung

Um die SQL Batchdatei auszuführen, selektieren Sie die Option **Datenbank→Ausführen** im Hauptmenü oder Klicken Sie auf den Button  . **Tun DB Script** wird die SQL Kommandos der Reihe nach an die Datenbank, die mit der ausgewählten Datenquelle korrespondiert, weiterleiten. Bei einem Fehler wird **Tun DB Script** stoppen und eine Fehlermeldung ausgeben.

➢ Verbindungsabbruch mit der Datenquelle

Nach der Ausführung muß die Datenquelle durch Klicken auf den Button  oder durch Anwahl der Option **Datenbank→Verbindung schließen** im Hauptmenü abgebaut werden. Ein Verbindungsabbruch wird ebenfalls durch Verlassen der Applikation bewirkt.

Hinweis:
Obwohl der erstrangige Zweck von **Tun DB Script** darin besteht, die **Tun SQL** Beispieldatenbank herunterzuladen, hat es auch noch weiteren Nutzen. Tatsächlich kann **Tun DB Script** eine ganze Liste von SQL Kommandos zur Erzeugung anderer Datenbanken, zur Aktualisierung oder Löschung einiger extrem großer Tabellen dienen.

Erzeugung einer virtuellen Datenquelle

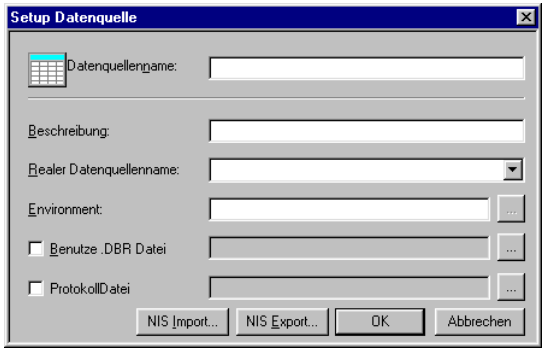
Tun DB Revamp kann, angepaßt an die Benutzerumgebung, virtuelle Tabellen an eine reale Datenbank linken. Weitere Informationen zu dieser Anwendung finden Sie im Abschnitt "**Tun DB Revamp**".

Wenn Sie eine virtuelle (revamped) Datenbank erzeugen, können Sie dafür Datenquellen erzeugen, als sei es eine reale Datenbank. Anwender können dann die mit **Tun SQL** gelieferten virtuellen ODBC Treiber benutzen, um auf die speziell für sie erzeugte virtuelle Datenbank zuzugreifen.

Eine virtuelle Datenquelle kann als die Verbindung einer realen Datenquelle mit einem Environment betrachtet werden.

Für die Erzeugung einer virtuellen Datenquelle greifen Sie auf die Konfiguration der virtuellen Datenquelle im **ODBC Administrator** zu. Wählen Sie den Treiber **Tunmap32**. Siehe auch "**Erzeugung einer Datenquelle** ".

Die folgende Dialogbox öffnet sich:



Data source name

Geben Sie den Namen der revampten Datenquelle ein.

Beschreibung

Geben Sie eine erklärende Beschreibung der Datenquelle ein.

Realer Datenquellenname

Geben Sie den Namen der Datenquelle für die reale Datenbank, von der die virtuelle Datenbank abhängt, ein.

Environment

Geben Sie den Namen eines Environments ein, für das Sie eine virtuelle Datenquelle erzeugen. Ein Environment ist ein Satz virtueller Tabellen: Jede revampte Datenbank kann ein oder mehrere Environments haben.

Klicken Sie den Button ..., um das Environment aus der Liste der in der Datenbank definierten Environments zu wählen.

Lokale .DBR Datei

Anstelle der Eingabe eines realen Datenbanknamens können Sie ein Environment aus einer lokalen .dbr Datei auswählen. Informationen zur Verwendung dieses Dateityps entnehmen Sie bitte dem Abschnitt "**Tun DB Revamp**".

Markieren Sie die Checkbox **Lokale .DBR Datei**. Geben Sie den vollständigen Pfadnamen der Datei an oder klicken Sie den Browse-Button  , um die "**.dbr**" Datei zu wählen. Dann wählen Sie das Environment (Feld: **Environment**).

Protokoldatei

Um Ihre SQL-Abfragen nachzuverfolgen, markieren Sie die Checkbox **Protokolldatei**. Die Speicherung erfolgt in einer Logdatei, die Sie mit einem Texteditor öffnen können. Hiermit können Sie nachvollziehen, was der ODBC Treiber macht, wenn Sie eine SQL-Abfrage an ihn übergeben.

Klicken Sie den Browse-Button  , um das Verzeichnis zu wählen, in dem die Logdatei gespeichert werden soll und geben Sie auch einen Dateinamen ein.

Konvertierungstabellen

Dieser Abschnitt kann beim ersten Lesen ausgelassen werden.

➢ Unterschiedliche Notation verschiedener Computersysteme

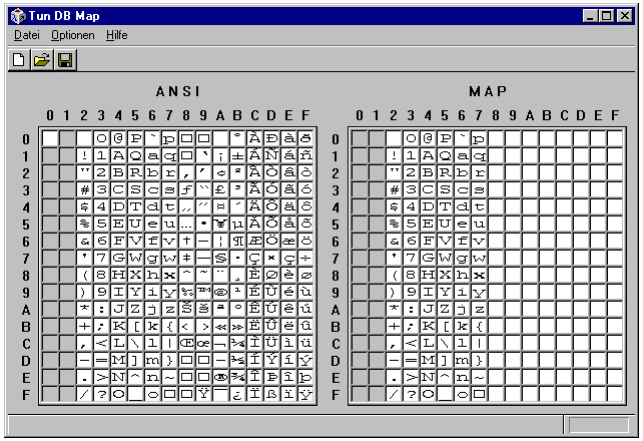
Während die Zeichen, die im englischen verwendet werden in der ASCII Tabelle (0-127) perfekt kodiert sind, ist dies für die Sonderzeichen anderer Sprachen (z.B. Französisch, Deutsch, Spanisch, Italienisch). Obwohl gewisse Standards existieren (z.B., werden diese nicht von jedem Computersystem unterstützt.

Da **Tun SQL** einen PC in die Lage versetzt, auf Daten zuzugreifen, die auf anderen Systemen (DBMS unter UNIX) gespeichert sind, wurde es als notwendig angesehen, einen speziellen Mechanismus in **Tun SQL** einzubauen, um unterschiedliche Notationen zwischen den verschiedenen Systemen zu berücksichtigen. Dieser Mechanismus ermöglicht es, Akzentzeichen (z.B. ein **é**) anzuzeigen, obwohl diese auf dem DBMS unter UNIX anders kodiert sind.

➢ Anlegen von Konvertierungstabellen

Tun SQL nutzt Konvertierungstabellen, die mit der Applikation **Tun DB Map** angelegt oder geändert werden können.

Starten Sie das Programm durch Klicken auf das **Tun DB Map** Symbol in der **Data Access** Gruppe (Menü **Start**, **Programme**, **Esker Tun** in Windows 95/98/2000 und Windows NT).



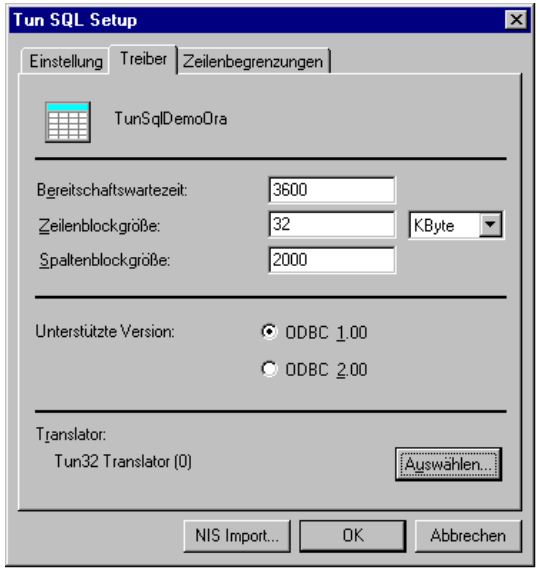
Die linke Tabelle zeigt alle auf dem PC verfügbaren Zeichen an (ASCII und erweiterter ASCII Code). Die rechte Tabelle sollte die gleichen Zeichen enthalten, jedoch an Positionen, die der Notation des DBMS auf der UNIX Maschine entsprechen. In der rechten Tabelle sind die ersten 128 Zeichen bereits enthalten, da sie auf beiden Systemen identisch sind.

Um ein bestimmtes Zeichen der rechten Tabelle zuzuweisen, wählen Sie ein Zeichen auf der linken Seite und "ziehen" es mit der Maus an die gewünschte Position in der rechten Tabelle.

Über die Hauptmenüoption **Datei** können weitere Konvertierungstabellen angelegt werden, die mit dem Suffix "**.ttt**" gespeichert werden.

➢ Einbindung von Konvertierungstabellen

Für ihren Einsatz müssen Konvertierungstabellen mit einer Datenquelle verbunden werden. Dies geschieht über die unten dargestellte Dialogbox; Details können im Abschnitt Übersetzung angegeben werden:



oder klicken Sie auf **Auswählen...**, um den ODBC Übersetzer zu wählen.

3

C-ISAM

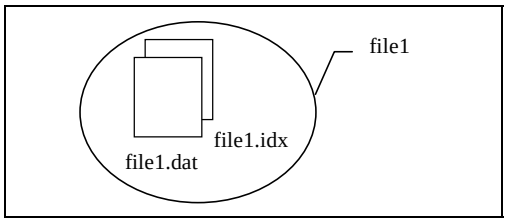
Einführung in C-ISAM

➢ Das C-ISAM Dateiablagesystem

C-ISAM (Indexed Sequential Access Method) ist eine von Informix entwickelte Bibliothek von C-Funktionen. Sie ermöglicht das Verwalten von indizierten sequentiellen Dateien (Dateierstellung sowie Einfüge-, Lösch- und Leseoperationen). C-ISAM enthält andere Merkmale wie Dateisperre und Transaktionsunterstützung zur Sicherung der Datenintegrität. Diese Merkmale stellen sicher, daß die Daten zugreifbar und gültig sind und richtig verwendet werden.

C-ISAM verwendet Datentypen ähnlich der in C verwendeten. Da C-ISAM diese Typen unabhängig vom verwendeten UNIX-System anwendet, kann die Art der Datenspeicherung sich von der Art der Darstellung während des Ausführens unterscheiden. C-ISAM enthält Konversionsfunktionen, die Runtime-Datenformat in Speicher-Datenformat umwandeln.

Eine C-ISAM-Datei ist in der Praxis eine Kombination von zwei Dateien. Eine Datei (file.dat) enthält Daten, eine andere einen Index, um die Daten in der Datendatei zu finden (file.idx). Diese beiden Dateien werden immer zusammen als eine logische C-ISAM-Datei verwendet.



➢ RDBMSs und C-ISAM

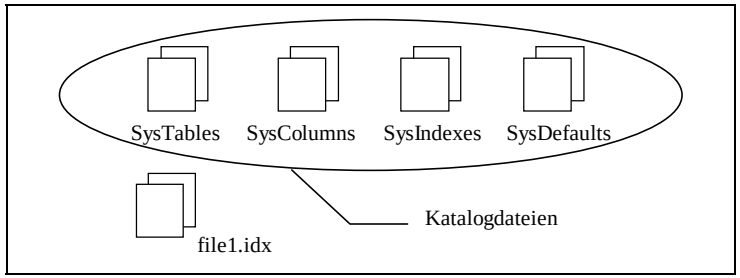
C-ISAM-Dateien verwenden indizierten sequentiellen Zugriff; Entwickler müssen daher die Dateistruktur verstehen und die Indexdatei für den Zugriff auf die Datendatei verwenden.

Erstellen von Datenbanksystemen mit C-ISAM-Dateien befreit die Entwickler von diesen Einschränkungen durch Einsatz einer in einem Katalog als Tabellen, Spalten und Indizes indizierten Dateistruktur.

➢ Tun SQL C-ISAM

C-ISAM verwendet C-Funktionen, um sequentielle Dateien abzufragen und zu aktualisieren. Der mit Tun SQL verschiffte C-ISAM-Treiber zeigt Ihnen sequentielle Dateien als eine normale relationale Datenbank mit Tabellen, Feldern und Schlüsseln. Auf diese Weise können Sie Standard-SQL-Befehle verwenden, um eine mit C-ISAM-Dateien erstellte Datenbank abzufragen oder zu aktualisieren. Der C-ISAM-Treiber übersetzt die Befehle in C-Funktionen, die die notwendigen Operationen auf die sequentiellen Dateien anwenden.

Um von einer sequentiellen Datenansicht in eine relationale Datenbank zu wechseln, fügt der C-ISAM-Treiber den Standard C-ISAM Daten- und Indexdateien beschreibende Datenbank-Dateien hinzu, den sogenannten **Katalog**. Es sind dies Dateien des C-ISAM-Typs (**.dat**-Datendateien und **.idx**-Indexdateien): SysTables, SysColumns, SysIndexes und SysDefaults.



sqltools ist ein UNIX-Tool für Erstellen und Verwalten von auf C-ISAM beruhenden Datenbanken. Es funktioniert nach dem gleichen Prinzip: SQL-Instruktionen werden für den Aufbau der Datenbank verwendet und in C-Funktionen übersetzt, bevor Sie das C-ISAM-Dateisystem lesen kann.

➢ C-ISAM-Treiber installieren

Um **Tun SQL C-ISAM** zu installieren, lesen Sie im Tun Installationshandbuch nach.

Sqltools verwenden

➢ Sqltools verwenden

Sie können **sqltools** online oder von einer semi-grafischen Oberfläche (Windows, Menüs) verwenden.

Verbinden Sie zum UNIX-Server, der C-ISAM-Dateien enthält. Wir empfehlen, daß Sie eine User Login-ID für den Zugriff auf C-ISAM-Dateien erstellen.

Wechseln Sie in das **sqltools** Installationsverzeichnis. Rufen Sie die Anwendung durch Eingabe folgender Befehle auf:

`sqltools`
für online-Anwendung
oder
`sqtools -v`
um die grafische Oberfläche zu verwenden.

➢ Datenbank erstellen

Der erste Schritt ist das Erstellen der Datenbank, die die Daten der C-ISAM-Dateien enthält. Dazu verwenden Sie eine der folgenden Methoden:

- Wählen Sie **Database➔Create** und geben Sie den Namen der zu erstellenden Datenbank ein.
- Geben Sie unten im Fenster (Eingabefenster) ein:

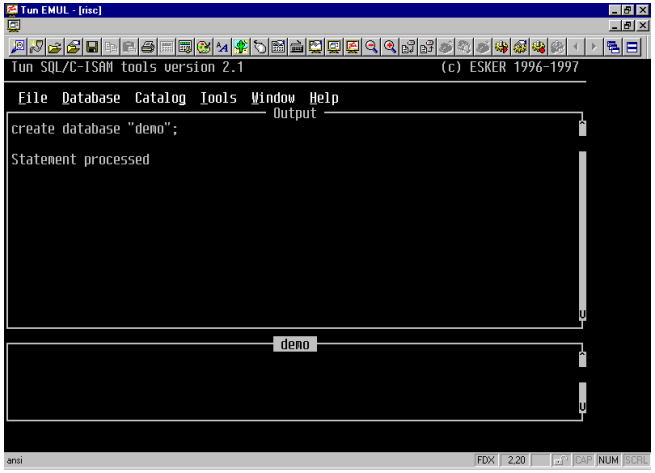
```
create database "databasename";
```

Damit generieren Sie ein Verzeichnis mit dem Namen der Datenbank und der Erweiterung **.ism**; die Variable ISAM-PATH bestimmt, wo das Verzeichnis generiert wird. Mehr Information über die ISAM-PATH Variable finden Sie im Abschnitt "C-ISAM-Treiber installieren" in der Tun Installationsanleitung. ".

Beispiel:
`Create database "demo";`
generiert das Verzeichnis directory demo.ism im Verzeichnis ***/TunSql/bases if ISAM-PATH=/TunSql/bases.***

Dieses Verzeichnis enthält die C-ISAM .dat- und .idx-Dateien, die die Datenbank beschreiben: SysTables, SysColumns, SysIndexes und SysDefaults, insgesamt vier logische C-ISAM-Dateien und acht Betriebssystem-Dateien.

Unten im Fenster (Eingabefenster) wird der Name der Datenbank eingegeben.



Hinweis:
Sie können Shell-Befehle direkt von **sqltools** ausführen. Stellen Sie dem Befehl ein Rufzeichen voran und schließen Sie ihn mit einem Strichpunkt ab.
Beispiel:
*! **ls -a** ;*

Für die Eingabe von Großbuchstaben verwenden Sie Anführungszeichen.
Beispiel:
!ls "/TunSql/locisam";

➢ Zu einer bestehenden Datenbank verbinden

Verwenden Sie eine bestehende Datenbank, können Sie Operationen auf Tabellen nach dem Verbinden durchführen.

Dazu wählen Sie **Database➔Connect** und geben Sie den Namen der Datenbank im Eingabefenster (unten im Fenster) ein.

➢ Tabellen erstellen

Sobald die Datenbank erstellt ist (das .**ism**-Verzeichnis enthält die beschreibenden Dateien der Tabelle), können Sie in ihr Tabellen erstellen. .

Sie können diesen Schritt mit einem Paar bestehender C-ISAM-Dateien (der **.dat**.-Datendatei und der **.idx**-Indexdatei) ausführen oder neue Dateien erstellen. Im zweiten Fall müssen Sie eine andere Anweisung und andere Vorsichtsmaßnahmen verwenden. Verwenden Sie die Anweisung **create table**, um das C-ISAM-Dateiepaar zusammen mit der Tabelle zu erstellen, die Anweisung **define table**, um eine Tabelle von zwei bestehenden C-ISAM-Dateien zu erstellen.

Tabellen neu erstellen

Um ein C-ISAM-Dateipaar zusammen mit der Tabelle zu erstellen, geben Sie die folgende Anweisung in das Eingabefenster ein; es hat nun den Namen der Datenbank in der Titelleiste:

```
create table tabellenname (field1 type1, field2 type2,...,
primary key(field1)));
```

Diese Anweisung generiert in der Datenbank eine Tabelle namens "tabellenname". Die Tabelle enthält die Felder field1, field2, etc. mit den Typen type1, type2, etc. Siehe die Typenliste.
Die C-ISAM-Daten- und Indexdateien werden im Verzeichnis der Datenbank erstellt (**databasename.ism**). Die Namen dieser Dateien sind die ersten sieben Zeichen des Tabellennamens und eine eigene Identifikationsnummer, die automatisch vergeben wird. Die Datendatei erhält die Erweiterung **.dat**, die Indexdatei **.idx**. Enthält der Dateiname weniger als sieben Zeichen, werden die Namen dieser zwei Dateien mit Unterstreichungen ("_") ausgefüllt.

Beispiel:
```
create table table1 (field1 longint, field2 char(25),
filler char (30), primary key(field1));
```

*erstellt die Dateien **table1_100.dat** und **table1_100.idx**. Die Tabelle enthält das Feld **field1** des Typs **longint**, das Feld **field2** enthält maximal 25 Zeichen des Typs **char** und das Feld **filler** enthält maximal 30 Zeichen des Typs **char**. Der primäre Schlüssel für diese Tabelle ist **field1**.*

Tabellen von einem bestehenden C-ISAM-Dateipaar erstellen

Um eine Tabelle in einer Datenbank zu erstellen, für die das C-ISAM-Dateipaar **.dat** und **.idx** bereits bestehen, geben Sie die folgende Anweisung in das Eingabefenster ein; der Name der Datenbank erscheint in der Titelleiste:

```
define table tabellenname file is filename (field1
type1, field2 type2,..., primary key(field1)));
```

Die Anweisung erstellt aus den bestehenden Dateien **filename.dat** und **filename.idx** eine Tabelle mit dem Namen "tabellenname".

Wichtiger Hinweis:
Bevor Sie die Tabelle verwenden (also bevor Sie zum Beispiel die Anweisung **select** auf die Tabellen anwenden), müssen Sie die durch **file is** definierten Dateien in das Datenbankverzeichnis kopieren.
Die Anweisung **define** kann jedoch auch ausgeführt werden, ohne daß die Dateien in dieses Verzeichnis kopiert wurden. Wollen Sie einen Index für die Tabelle erstellen, ist es sogar empfehlenswert, diese Dateien erst **nach** dem Ausführen von **create index** zu kopieren.

➢ Index erstellen

Um einen Index für die ersten acht Spalten einer Tabelle zu erstellen, geben Sie die folgende Anweisung in das Eingabefenster ein; der Name der Datenbank erscheint in der Titelleiste:

```
create unique index indexname on tabellenname
(field1, field3);
```

Diese Anweisung erstellt in der Tabelle "tabellenname" den Index "indexname" für Spalten **field1** und **field3**.

➢ C-Strukturen

Jede an **sqltools** übergebene Anweisung wird für das C-ISAM-Dateisystem in C-Code übersetzt. Um die entsprechende C-Struktur für eine Tabelle zu sehen, wählen Sie **Catalog➔GetCStruct**.

Die Anweisung, die **table1**erstellt, generiert zum Beispiel die folgende C-Struktur:

```
struct root_table1              /* file "table1_100" */
 {long          lint_field1;    /* field1 longint */
 char           chr_field2[25]; /* field2 char(25) */
 char           chr_filler[30]; /* filler char(30) */
 unsigned char  null_flags[1];  /* reserved */
};
```

In diesem Beispiel gibt es ein reserviertes Feld (**unsigned char**) mit einer Länge von einem Byte. Dieses Feld ist spezifisch für das C-ISAM Managementsystem. Wird eine Tabelle mit der Anweisung **create table** erstellt, fügt **sqltools** der Tabelle dieses reservierte Feld automatisch hinzu.

➢ Validieren einer von bestehenden C-ISAM-Dateien erstellten Tabelle.

Erstellen Sie (mit der Anweisung **define**) eine Tabelle von einem bestehenden C-ISAM-Dateipaar, müssen Sie darauf achten, daß die erstellte Tabellestruktur der Struktur der C-ISAM-Dateien entspricht.

Sie erstellen zum Beispiel die Tabelle "table2", beruhend auf den in C geschriebenen C-ISAM-Dateien **filename.dat** und **filename.idx**. Diese Dateien haben die folgende Datensatzstruktur:

- Ein Feld bestehend aus einer longint-Variablen,
- ein Feld bestehend aus einem Bereich von 25 char-Variablen,
- ein Feld bestehend aus einem Bereich von 30 char-Variablen.

Sie definieren "table2" wie folgt:

```
define table table2 file is table1_100 (field1
longint, field2 char(25), filler char(25));
```

Die Datensätze dieser Tabelle haben die folgende Struktur:

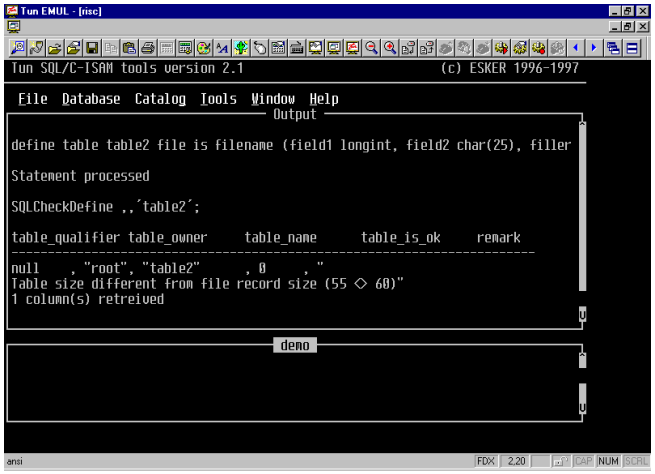
- Ein Feld longint,
- zwei Bereichsfelder mit einer Länge von 25 Zeichen**.**

Diese Struktur entspricht nicht der Struktur der C-ISAM-Dateien, auf denen die Tabelle beruht.

In diesem Fall besteht keine Übereinstimmung zwischen der erstellten Tabelle und den C-ISAM-Dateien, auf denen Sie beruht.

Um zu verifizieren, daß die Datenstruktur der Tabelle derjenigen der ursprüngliche C-ISAM-Dateien entspricht, wählen Sie **Catalog➔CheckDefine**. Um alle Tabellen der Datenbank zu verifizieren, wählen Sie **Tools➔Check Catalog**.

In unserem Beispiel zeigt das Ausgabefenster die folgenden Ergebnisse:



Es erscheint die folgende Meldung:

```
Table size different from file record size (55 <>
60)"
```

Diese Meldung warnt, daß Tabelle **table2** anders definiert wurde, als die C-ISAM-Dateien "fileneame", auf den Sie beruht. Die Anweisung **define** sollte lauten wie folgt:

```
define table table2 file is filename (field1 longint,
field2 char(25), filler char(30));
```

➢ Die C-Struktur einer Tabelle betrachten

Sie können die C-Struktur einer Tabelle sehen, also die verschiedenen Spalten, die erstellt wurden (Name, Datentyp und Länge) sowie Datenmanagement betreffende Information (z.B. den primären Schlüssel).

Dazu wählen Sie **Catalog➔GetCStruct** und geben Sie den Namen der Tabelle ein, deren C-Struktur Sie zu sehen wünschen. Um die C-Strukturen aller Tabellen in der Datenbank zu sehen, wählen Sie **Tools➔List Structures**.

Das untenstehende Beispiel zeigt das Resultat dieses Befehls für eine mit einem primären Schlüssel erstellten Tabelle.

Die nachfolgenden Anweisungen erstellen die Tabelle:

```
create table customer
(cust_number longint,
 cust_name char(20),
 cust_address1 char(20),
 cust_address2 char(20),
 filler char(20),
 primary key (cust_number)
);
```

Die generierte C-Struktur ist:

```
struct doc_customer            /* file "custome110" */
{long    lint_cust_number;     /* cust_number longint */
 char    chr_cust_name[20];    /* cust_name char(20) */
 char    chr_cust_address1[20];/* cust_address1 char(20) */
 char    chr_cust_address2[20];/* cust_address2 char(20) */
 char    chr_filler[20];       /* filler char(20) */
 unsigned char null_flags[1];  /* reserved */
};

struct keydesc idx_customer_1;
idx_customer_1.k_flags = ISNODUPS;
idx_customer_1.k_nparts = 1;
idx_customer_1.k_part[0].kp_start = 0;
idx_customer_1.k_part[0].kp_leng  = 4;
idx_customer_1.k_part[0].kp_type  = LONGTYPE;
```

Der erste Teil des Codes zeigt die Struktur der Tabelle (die Felder), der zweite den primären Schlüssel und seine verschiedenen Zuordnungen.

➢ Kataloginformation

Mit Hilfe des Menüs **Catalog** erhalten Sie folgende Informationen über den Katalog:

- **TypeInfo**: Jeder Datentyp wird entsprechend dem ODBC-Standard durch eine Nummer identifiziert. Geben Sie die Nummer des zu verifizierenden Datentyps ein, oder 0, um die Liste der Dateitypen zu sehen.
- **Tables**: Sie können den Katalog nach Information über seine Tabellen abfragen. Verwenden Sie dazu den Namen der Anwenders, der die Tabelle erstellt hat, den Namen der Tabelle oder den Tabellentyp (Systemtabelle, Synonym, etc.). Geben Sie % ein, damit Ihre Abfrage alle Tabellen oder Tabellentypen abdeckt.

- **Columns**: Sie können den Katalog nach Information über seine Spalten abfragen. Verwenden Sie dazu den Namen der Anwenders, der die Tabelle erstellt hat, den Namen der Tabelle oder den Spaltenname. Geben Sie % ein, damit Ihre Abfrage alle Tabellen oder Spalten abdeckt
- **Statistics**: Sie können über die Daten im Katalog statistische Angaben erhalten.
- **PrimaryKeys**: Sie können den Katalog nach den Primärschlüsseln seiner Tabellen abfragen. Verwenden Sie dazu den Namen des Anwenders, der die Tabelle erstellt hat, oder den Namen der Tabelle. Geben Sie % ein, damit Ihre Abfrage alle Tabellen abdeckt

Weitere Information über die Optionen im Menü **Catalog** finden Sie im entsprechenden Abschnitt im ODBC Standardhandbuch.

➢ Tabelle bearbeiten

Die Länge der Datensätze in einer Tabelle wird beim Erstellen der Tabelle festgelegt. Sie können daher keine Datensätze hinzufügen oder löschen, falls das die Gesamtlänge beeinflußt.

Wollen Sie die Struktur einer Tabelle ändern, müssen Sie zuerst die Tabelle löschen, unter Anwendung der in "Tabelle löschen" beschriebenen Vorsichtsmaßnahmen.

➢ Tabelle löschen

Sie haben zwei Möglichkeiten, eine Tabelle von einem Katalog zu löschen.

Sie haben die Tabelle mit der Anweisung **create table** erstellt: Verwenden Sie **drop table**, um Sie zu löschen. **Hinweis**: Dieser Befehl entfernt alle Referenzen zur Tabelle von den Katalogdateien und löscht das mit der Tabelle verknüpfte C-ISAM-Dateipaar.

Sie haben die Tabelle mit der Anweisung **define table** erstellt: Verwenden Sie **undefine table**, um den **define**-Befehl rückgängig zu machen. Dieser Befehl entfernt nur alle Referenzen zur Tabelle von den Katalogdateien.

Hinweis:
Sie können **drop table** für eine Tabelle verwenden, die Sie mit **define table** erstellt haben. **Drop table** löscht jedoch das durch **file is** definierte C-ISAM-Dateipaar, falls Sie sich im Datenbankverzeichnis befinden. Haben Sie die Tabelle von bestehenden C-ISAM-Dateien erstellt, sollten Sie daher beim Einsatz des Befehls **drop table** sehr vorsichtig sein.

Wollen Sie **drop table** verwenden, jedoch das zu der zu löschenden Tabelle gehörende C-ISAM-Dateipaar behalten, müssen Sie diese zwei Dateien zuerst in ein anderes Verzeichnis kopieren. Nach dem Löschen der Tabelle können Sie dann diese Backup-Kopien verwenden.

➢ Wartung der C-ISAM-Dateien

Erstellen Sie Tabellen von bestehenden C-ISAM-Dateien (mit **define table**), müssen Sie diese Dateien in das Verzeichnis der Datenbank kopieren, falls Sie die Tabellen verwenden wollen.

Werden diese Dateien jedoch von anderen Anwendungen verwendet und aktualisiert, ist es vorteilhaft, diese Änderungen in Ihrer C-ISAM-Datenbank verwenden zu können. Dazu dürfen Sie die Dateien nicht kopieren, sondern müssen sie von der Datenbank symbolische Verknüpfungen zu den bestehenden C-ISAM-Dateien machen.

Beispiel:
*Sie erstellen im Verzeichnis **/TunSql** die Datenbank **dbtest**. Für die Tabellen in dieser Datenbank verwenden Sie die Dateien **filename.dat** und **filename.idx** im Verzeichnis **/data**; diese Dateien werden von anderen Anwendungen verwendet.*

*Sie erstellen im Verzeichnis **/TunSql/dbtest.ism** (dem Datenbankverzeichnis) symbolische Verknüpfungen zu diesen Dateien. Dazu verwenden Sie folgenden Befehl:*

```
ln -s /data/filename.* /TunSql/dbtest.ism
```

➢ Datenbank löschen

Um eine Datenbank zu entfernen (löschen), wählen Sie **Database➔Drop** und geben Sie den Namen der zu löschenden Datenbank ein, oder geben Sie folgenden Befehl in das Eingabefenster ein:

```
drop database databasename
```

➢ Ergebnisse speichern

Sie können die im oberen Fensterteil (Ausgabefenster) angezeigten Ergebnisse in einer Textdatei mit der Erweiterung **.res** speichern.

Dazu wählen Sie **File➔Save as...** und bestimmen Sie das Verzeichnis und den Namen der zu speichernden Datei.

➢ Skript ausführen

Verwenden Sie die von **sqltools** unterstützte SQL-Aweisung, so können Sie mit der Option **File➔Execute** ein SQL-Skript ausführen.

TEIL 2
DATENBANK REVAMPING

4

REVAMPING

Virtuelle Datenbanken

Der Großteil der heutigen strukturierten Datenspeicherung besteht aus Relational Database Management Systems RDBMS. Datenbanken können die Unternehmensdaten speichern und durch Applikationen aktualisiert werden. Die Masse von in dieser Weise gesammelten Daten sind auch für eine große Zahl von Anwendern von Interesse, die aus dem Datenreservoir Informationen für ihre Arbeit schöpfen können (Leistungsanzeigen, Statistiken, Expertensystem). Die SQL Sprache wird zur Aktualisierung und Abfrage von Datenbanken benutzt.

Die Struktur von Datenbanken, die sich im Herzen des Informationssystems befinden, können jedoch den Zugriff auf die enthaltene Information auf jeder Unternehmensebene erschweren:

- Es gibt eine für den Durchschnittsanwender zu große Anzahl von Tabellen und Datenfeldern in einer Datenbank, da er immer nur an einem Teil der Daten interessiert ist.
- Datenbankstrukturen sind naturgemäß komplex und erfordern ein hohes Maß an Erfahrung, um damit sachgemäß umgehen zu können.
- Die Rechenumgebung von Datenbanken ist nicht sehr anwenderfreundlich. Beispielsweise werden die Namen von Tabellen und Feldern selten klartextlich benannt.
- Datenmanipulation und -nutzung erfordern die Kenntnis der SQL Sprache, um Datenbanken abzufragen und an die gewünschten Resultate zu kommen.

Verschiedene Anstrengungen zur Überwindung dieser Hindernisse und Erleichterung des Zugriffes auf Datenbanken (z. B. durch Einbau grafischer Schnittstellen in Datenbankabfragetools) sind unternommen worden.

Der nächste Schritt zielt auf die Befreiung des Anwenders von der Notwendigkeit des technischen Verstehens von Datenbanken, indem ihm nur die für ihn notwendigen Informationen in der für seine Arbeitumgebung bequemsten Art und Weise verfügbar gemacht werden.

Die Konsequenzen eines solchen Wechsels sind:

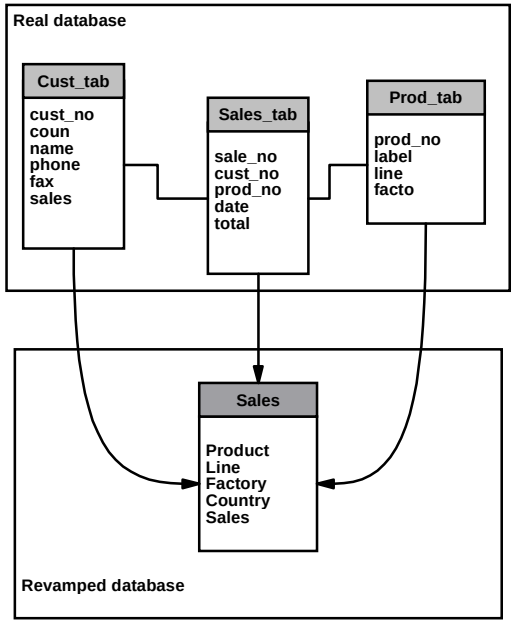
- Verbesserte Produktivität: der Endanwender wird autonom in seiner Datennutzung, Analyse und Entscheidungsprozeß kosten weniger Zeit, da sie leichter fallen.
- Relevantere Informationen: da er nur noch die tatsächlich benötigten Daten, die er auch bewältigen kann, erhält, kann der Anwender seine Analyse und Synthese präzisieren und seine Resultate verfeinern.

➢ Revamping

Der Grundsatz des Revampings besteht in der Konstruktion einer an die Anwenderbedürfnisse angepaßten virtuellen Datenbank aus einer existierenden Datenbank. Obwohl sie nicht als wirkliche Datenbank existiert, wird die neue Struktur vom Anwender als normale Datenbank wahrgenommen, deren Tabellen und Felder jedoch exakt seinen Bedürfnissen entsprechen: die Datenbank enthält nur jene Informationen, die der Benutzer tatsächlich für seine Analysen braucht, in einer Form, die seinen Erfordernissen gerecht wird (verständliche Datennamen, vordefinierte Funktionen).

Die redefinierte Datenbank wird von einem Adminstrator, der die Tabellen und Felder aus wirklichen Datenbanken rekonfiguriert, zusammengebaut.

Zum Beispiel:



Das obige Beispiel (das im übrigen die international gebräuchlichen englischen Kurzbezeichnungen enthält) besteht die wirkliche Datenbank aus drei Tabellen: "Cust_tab" (Kundentabelle), "Sales_tab" (Vertriebstabelle) und "Prod_tab" (Produkttabelle).

Der Administrator definiert eine virtuelle Tabelle, die die Verkaufsergebnisse pro Produkt, Produktlinie, Herstellungsort und Land darstellt.

Die virtuelle Tabelle mit dem Namen "Sales" enthält folgende Felder:

- Product (wirkliches Feld: "prod_tab.label»)
- Line (wirkliches Feld: "prod_tab.line")
- Factory (wirkliches Feld: "prod_tab.fact")
- Country (wirkliches Feld: "cust_tab.coun")
- Sales (wirkliches Feld: "sales_tab.total")

Die virtuelle Tabelle erstellt eine Verknüpfung zwischen den Tabellen "Prod_tab" und "Sales_tab" über das gemeinsame Feld "prod_no" und eine Verknüpfung zwischen den Tabellen "Sales_tab" und "Cust_tab" über das gemeinsame Feld "cust_no".

Revamping in Tun SQL

Tun SQL kann die Verwaltung und Nutzung virtueller Datenbanken auf Grund der folgenden zwei Komponenten gewährleisten:

- Dem Datenbankadministrator von **Tun DB Revamp**, der das Revamping verwaltet.
- Dem virtuellen ODBC Treiber, der dem Anwender den Zugriff auf durch den Adminstrator revampte Datenbanken gestattet.

➢ Der Tun DB Revamp Administrator

Ziel der virtuellen **Tun SQL** Datenbank ist es, dem Anwender kontextsensitiv umdefinierte Information für eine bestimmte „Umgebung“ anzubieten.

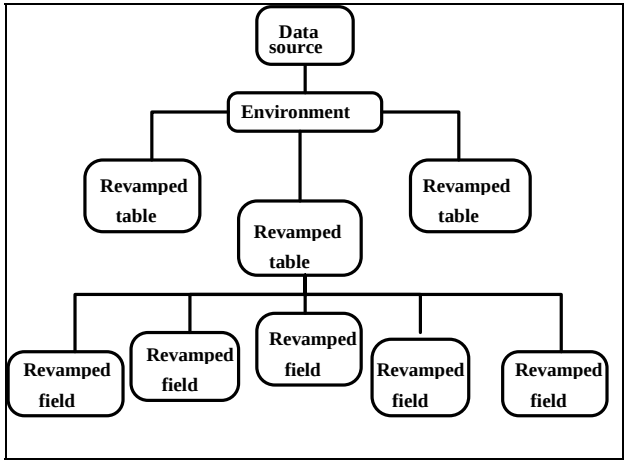
Dank einer intuitiven grafischen Schnittstelle kann der Administrator soviele "Umgebungen" für unterschiedliche Benutzer oder Benutzertypen definieren wie er will. Die "Umgebung" basiert auf "Tätigkeit": ein Buchhalter soll nur die Daten für die Buchhaltung sehen, Vertriebsmitarbeiter nur die für ihr Haupttätigkeitsfeld zuständigen Tabellen.

Jede Umgebung kann über jedes ODBC Frontend, das die Abfrage auf eine spezielle Datenquelle macht, erreicht werden (vergleichen Sie dazu das Architekturdiagramm im Abschnitt **Virtueller ODBC Treiber**).

Das virtuelle Datenbankmodell sieht folgendermaßen aus:

- Eine oder mehrere Umgebungen, die aus einer echten Datenquelle durch Auswahl benutzerspezifischer Tabellen definiert werden.
- Die Tabellen in einer Umgebung sind entweder tatsächliche Tabellen der wirklichen Datenbank oder Verknüpfungen aus zwei oder mehr Tabellen (Vorstellung der Ansicht).
- Jede Tabelle enthält die für den Anwender erforderlichen Felder und nur diese.

- Die revampten Felder sind entweder existierende Felder aus wirklichen Tabellen oder berechnete Felder, die dem Endanwender den Gebrauch der Datenbank weiter erleichtern.
- Die revampten Tabellen und Felder können umbenannt werden, so daß sie für den Endanwender leichter verständlich sind (z. B. "Cust_tab" kann "Customer Table" (Kundentabelle) und "Cust_no" "Client Number" (Kundennummer) werden).



Die in einer Umgebung revampten Tabellen existieren in der Datenbank nicht physikalisch. Dagegen wird die revampte Datenbank in indizierter Form in drei ergänzenden Tabellen in der Datenbank erzeugt:

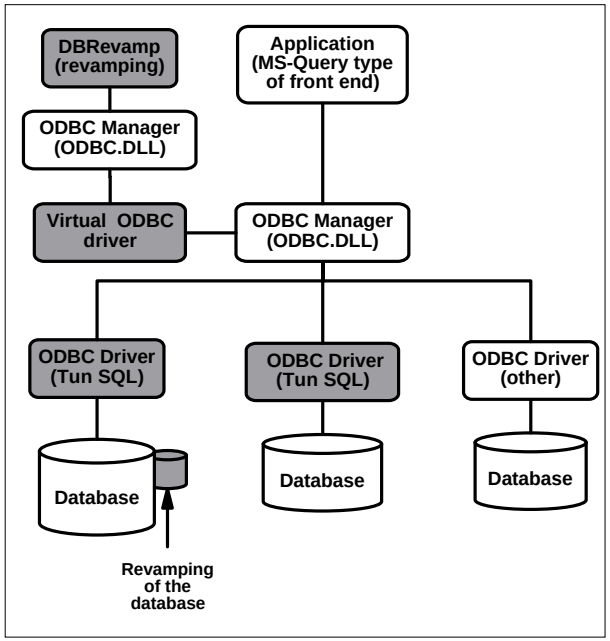
- Die Umgebungstabelle mit einer Liste von Umgebungen der Form "Umgebungsname" und "Beschreibung".
- Eine Tabelle mit der Liste umdefinierter Tabellen, jeweils durch einen Namen, eine Beschreibung und die Umgebung, zu der sie gehört, indiziert.
- Eine Tabelle umdefinierter Felder, die durch ihren Namen, eine Beschreibung, eine Herkunft (existierendes Feld, errechnete Daten, Datenzusammenfassung) und den Namen der virutellen Tabelle, zu dem es gehört, indiziert wird.

Nach einer Revamping-Operation wird eine Datenbank immer diese drei zusätzlichen Tabellen enthalten.

➢ Virtueller ODBC Treiber

Für den Endanwender stellt sich die Abfrage einer virtuellen Datenbank ähnlich dar wie die einer wirklichen Datenbank im nur lesenden Zugriff. Diese Transparenz wird durch einen speziellen in **Tun SQL** integrierten ODBC Treiber für virtuelle Datenbanken erzeugt.

Wenn der ODBC Manager (ODBC.DLL) Abfragen aus einer bestimmten Umgebung empfängt, reicht er sie an den virtuellen ODBC Treiber weiter, dessen Aufgabe dann darin besteht sie in formgerechte Anfrgaen für die wirkliche Datenbank zu übersetzen. Das Abfrageergebnis wird nach Rückübersetzung durch den virtuellen ODBC Manager an den normalen ODBC Treiber der Datenbank zurückgeschickt.



5

TUN DB REVAMP NUTZUNG

Dieses Kapitel enthält eine Beschreibung der grundsätzlichen **Tun DB Revamp** Befehle für allgemeinen Gebrauch.

Allgemeine Anmerkungen

➢ Sprache wählen

Um die von Ihnen gewünschte Sprache zu wählen, klicken Sie auf die Option **?→Sprache** und wählen Sie die zu benutzende Arbeitssprache.

➢ Änderung der Anzeige

Um die Kontrollfunktionen, die im Hauptfenster von **Tun DB Revamp** angezeigt werden, zu ändern,

- wählen Sie die Option **Ansicht→Werkzeuge** aus dem Hauptmenü oder brechen Sie diese ab, um das Toolbar sichtbar oder unsichtbar zu machen.
- wählen Sie die Option **Ansicht→Statuszeile** aus dem Hauptmenü oder brechen Sie diese ab, um die Statusleiste sichtbar oder unsichtbar zu machen.
- wählen Sie die Option **Ansicht→Eigenschaftenzeile** aus dem Hauptmenü, um die Eigenschaftenleiste sichtbar oder unsichtbar zu machen.

➢ Kopieren eines Objekts

Verwenden Sie eine der folgenden Optionen, um ein Objekt zu kopieren:

1. Um die "**drag and drop**" Methode zu verwenden, wählen Sie das zu kopierende Objekt und ziehen Sie die Maus zum Zielort, wobei Sie die linke Maustaste gedrückt halten.

2. Verwenden Sie die Hauptmeüoption **Bearbeiten→Kopieren**, um das selektierte Objekt zu kopieren und dann die Option **Bearbeiten→Einfügen**, um es am Zielort einzufügen.
3. Wählen Sie die Optionen **Kopieren** und **Einfügen** im Kontextmenü (das durch Klicken der rechten Maustaste angezeigt wird), um das selektierte Objekt zu kopieren und am Zielort einzufügen.
4. Benutzen Sie die Tastenkombinationen **Strg-C** (kopieren) und **Strg-V** (einfügen), um die Operation auszuführen.
5. Benutzen Sie die Werkzeugleisten-Buttons,  (kopieren) und  (einfügen), um die Operation auszuführen.

➢ Ein Objekt löschen

Um ein Objekt zu löschen, wählen Sie es aus und vollziehen Sie eine der folgenden Maßnahmen:

1. Verwenden Sie die Hauptmenüoption **Bearbeiten→Löschen**.
2. Benutzen Sie die Kontextmenüoption **Löschen**.
3. Benutzen Sie eine der **Entf** Tasten auf der Tastatur.
4. Klicken Sie auf den  Button in der Werkzeugleiste.

➢ Ein Objekt umbenennen

Um ein Objekt umzubenennen muß es zuerst ausgewählt werden. Danach sind folgende Methoden möglich:

1. Nutzung der **Allgemein** Tabelle in der Eigenschaftenleiste.
2. Nutzung der Funktionstaste **F2** und Ersetzung des alten durch den neuen Namen.
3. Erneutes Anklicken des Objekts und Fortsetzung wie bei Methode.2.

➢ Änderungen speichern

Um Änderungen, die an Eigenschaftswerten vorgenommen wurden, zu speichern, drücken Sie Enter, wobei der Cursor sich in der relevanten Dialogbox befinden muß, oder benutzen Sie den Button **Anwenden**.

➢ Hilfe erlangen

Um die Online-Hilfe zu aktivieren oder mehr Informationen über **Tun DB Revamp** zu erlangen, Klicken Sie auf die Hauptmenüoption **?→Über DBRevamp**, oder benutzen Sie den Werkzeugleisten-Button  .

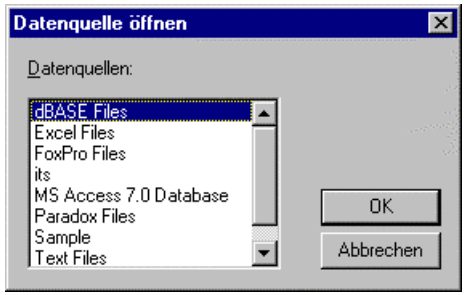
➢ Tun DB Revamp verlassen

Um die Applikation zu verlassen, klicken Sie die Option **Datei→Beenden**.

Umgebung einer Datenquelle importieren

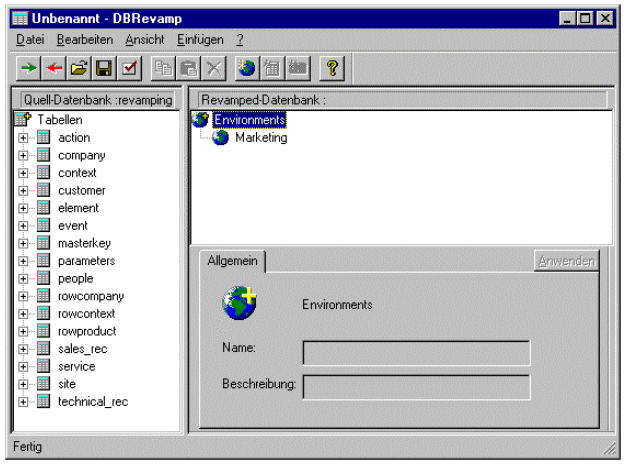
Um eine wirkliche Datenbank umzudefinieren (revampen), müssen Sie eine zugehörige Datenquelle auswählen. Sie tun das durch Anwahl der Option **Datei→Import Datenquelle** oder sonst durch Klicken auf den Button  in der Werkzeugleiste.

Die folgende Dialogbox wird angezeigt:



Angezeigt wird eine Liste von auf dem PC deklarierten Datenquellen. Zur Erzeugung einer Datenquelle sehen Sie bitte im Abschnitt "**Erzeugung einer Datenquelle**" nach. Da die virtuellen Datenbanken, die durch Revamping einer tatsächlichen Datenbank erreicht werden, nicht umdefiniert werden können, erscheinen Sie nicht in dieser Liste. In allen Windows-Anwendungen, die dem Endbenutzer zur Verfügung stehen (z. B. Microsoft Query), sind sie dagegen sehr wohl sichtbar.
Wählen sie die Datenquelle aus, die Sie nutzen wollen.

Ein **Tun DB Revamp** Fenster, das dem folgenden ähnlich ist, erscheint am Bildschirm:



Die Tabellen der wirklichen Datenbank erscheinen auf der linken Fensterseite.

Falls die fragliche Datenbank noch nicht mit **Tun DB Revamp** revampt worden ist, wird die rechte Fensterseite eine leere Umgebung namens "Neues Environment" enthalten, die Sie als erste Umgebung definieren können.

Wenn anderenfalls die Datenbank bereits mittels **Tun DB Revamp** bearbeitet worden ist (wodurch es sich um die Aktualisierung einer virtuellen Datenbank handelt), wird die rechte Fensterseite die Liste der bereits angelegten Umgebungen und deren Inhalte anzeigen.

Erzeugung eines Environment

Zur Definition eines neuen Environments für die ausgewählte Datenquelle wählen Sie die Environment-Root ("Environments" genannt) und wählen **Einfügen→Neues Environment** aus dem Hauptmenü. Sie können auch den Button  klicken.

Geben Sie einen Namen und optional eine Beschreibung für diese Umgebung ein.

Erzeugung einer virtuellen Tabelle

Um in einem Environment eine virtuelle Tabelle zu erzeugen, wählen Sie das Environment und gehen wie folgt vor:

- Wählen Sie **Einfügen→Neue Tabelle** aus dem Hauptmenü oder wählen **Neue Tabelle** aus dem Kontextmenü des Environments. Sie können auch den Button  nutzen.
- Wählen Sie **Ansicht→Eigenschaftenbox**, um die Eigenschaftenbox der neu erzeugten Tabelle anzeigen zu lassen (wenn dies noch nicht erfolgt ist).
- Im Register **Allgemein** in der Eigenschaftenbox geben Sie einen Namen und optional eine Beschreibung für die Tabelle ein. Sie können auch die **F2**-Taste benutzen, um eine Tabelle umzubenennen.

Erzeugung eines Feldes

In einer virtuellen Tabelle können Sie:

- Ein existierendes Feld aus einer realen Datenbank einfügen, ohne dessen Definition zu ändern.
- Ein neues virtuelles Feld aus den Feldern der realen Datenbank erzeugen.

➢ Existierende Felder

Sie können ein existierendes Feld aus einer Tabelle der wirklichen Datenbank direkt in die virtuelle Tabelle kopieren, indem Sie:

- Eine der in der Einführng beschriebenen Methoden benutzen. (**drag 'n drop**, **Kopieren/Einfügen**, Tastatur- Kurzbefehl und Werkzeugleisten-Button), um das Feld in der Tabelle der wirklichen Datenbank zu wählen und in der virtuellen Tabelle der umdefinierten Datenbank abzulegen.
- Falls gewünscht, können Sie den Feldnamen ändern und ihm in der entsprechenden **Allgemein** Tabelle eine Beschreibung anhängen (oder auch mit der **F2** Funktionstaste umbenennen).

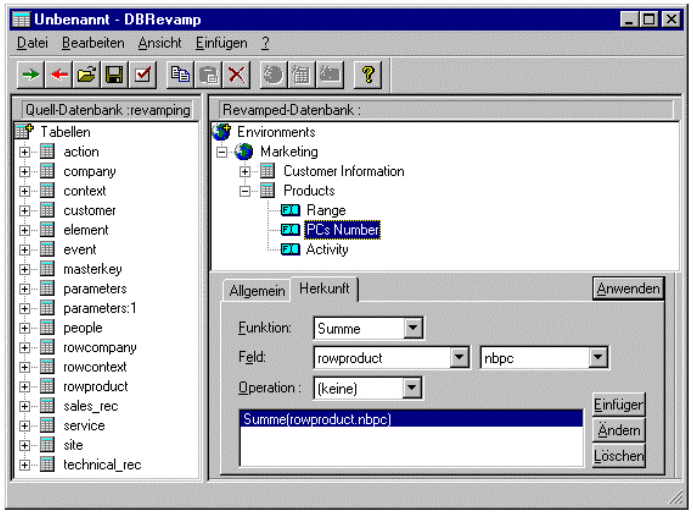
➢ Neues Feld

Um ein neues Feld in einer virtuellen Tabelle zu definieren, wählen Sie die virtuelle Tabelle und gehen wie folgt vor:

- Wählen Sie **Einfügen→Neues Feld** aus dem Hauptmenü oder **Neues Feld** aus dem Kontextmenü der Tabelle. Sie können auch den Button  nutzen.
- Wählen Sie **Ansicht→Eigenschaftenbox**, um die Eigenschaftenbox der neu erzeugten Tabelle anzeigen zu lassen (wenn dies noch nicht erfolgt ist).
- Geben Sie einen Namen und optional eine Beschreibung in der **Allgemein** Tabelle ein.
- Klicken Sie die Tabelle **Herkunft** an. Sie können dann:

    - Dem Feld eine Funktion hinzufügen: bestimmen Sie die Funktion Ihrer Wahl aus der Listbox **Funktion**. Die verfügbaren Funktionen sind: **Summe**, **Minimum**, **Maximum**, **Anzahl**, **Durchschnitt** oder **Keine**.
    - Einen Wert aus einem existierenden Feld einer wirklichen Datenbank dem neuen Feld oder der oben gewählten Funktion hinzufügen: wählen Sie die wirkliche Tabelle und das Feld aus den zwei Felder Listenboxen.
    - Eine Operation dem oben ausgewählten Feld hinzufügen: bestimmen Sie den gewünschten Operator in der **Operation** Listbox. Die verfügbaren Operationen sind: +, - , * , / , oder keine. Der Operator + kann zur Zusammenfassung von Zeichen benutzt werden.

- Klicken Sie dann den Button **Einfügen**, um diese Optionen der Felddefinition hinzuzufügen.



Beispiel1:
Sie haben Zugriff auf eine wirkliche Tabelle namens "res_tab", die vier Felder res1, res2, res3 und res4 entsprechend den vierteljährlichen Ergebnissen enthält. Sie wollen das Feld "Result" in einer Tabelle Ihrer virtuellen Datenbank, das die Summe der vier wirklichen Felder ausgibt, anlegen.

*In der Tabelle **Herkunft,** für das Feld "Result":*

- *Wählen Sie die Tabelle "res_tab" und das Feld "res1" in den **Feld** Options- Listboxen.*
- *Wählen Sie den Operator + in der Listbox der Option **Operation**.*
- *Klicken Sie auf den Button Ändern, um den Vorgabeeintrag zu ersetzen. Der neue Eintrag lautet "res_tab.res1 +".*
- *Als nächstes wählen Sie die Tabelle "res_tab" und das Feld "res2" in der Listbox der Option **Feld**.*
- *Wählen Sie erneut den Operator + aus der **Operation** Listbox.*
- *Klicken Sie auf den Button **Einfügen**, um den neu erstellten Eintrag "res_tab.res2 +" hinzuzufügen.*
- *Wiederholen Sie den Vorgang für das Feld "res3".*
- *Beim Feld "res4", wählen Sie den Operator **keine** statt +.*

- *Das Feld "Result" wird schließlich aus der folgenden Liste definiert:*

  res_tab.res1+
  res_tab.res2+
  res_tab.res3+
  res_tab.res4

  woraus sich ergibt, daß das Feld Result die Summe aus den Feldern "res1"," res2", "res3" et "res4" enthält.

Beispiel 2:
Sie möchten im Ergebnis-Feld die Summe der Jahresverkäufe anzeigen lassen. Dazu suchen Sie die Summe aller Felder res1, res2,... und die Summe dieser vier Resultate.

In der Tabelle ***Herkunft,*** *für das Feld "Result":*

- *Wählen Sie die Summenfunktion aus der* ***Funktion*** *Listbox.*
- *Wählen Sie die res_table Tabelle und das Feld "res1" aus jeder* ***Feld*** *Listbox.*
- *Wählen Sie den "+" Operator aus der* ***Operation*** *Listbox.*
- *Klicken Sie den* ***Mod*** *Button, um den Default Eintrag zu ersetzen. Der neue Eintrag ist "res_tab.res1 +".*
- *Gehen Sie genau so für die Felder "res2" and "res3" vor. Für das Feld "res4" wählen Sie den Operator* ***keine*** *anstelle des "+" Operators.*

Sie können einen Punkt in der Definition eines Feldes ändern, indem Sie den Button **Ändern** (das hervorgehobene Element ist durch die oben ausgewählten Werte ersetzt) bedienen. Klicken Sie den Button **Löschen**, um das hervorgehobene Element zu löschen.

Nach der Definition des Feldes, Klicken Sie den **Anwenden** Button, um die neuen Optionen zu aktivieren.

Sobald ein neues virtuelles Feld angelegt wurde, denken Sie an die Definition der Verknüpfungen zwischen den Tabellen, die gegebenenfalls bei der Erzeugung der virtuellen Tabelle benötigt wurden. Lesen Sie dazu den Abschnitt "**Tabellenübergreifende Verknüpfungen**".

Um zu überprüfen, ob die den erstellten Feldern zugewiesenen Rechenoperationen Ihren Wünschen entsprechen, verwenden Sie **Tun Dialogbox Revamp's** Abfragefunktion. Siehe "**Reale und wirkliche Datenbanken abfragen**".

Feldfilter zuweisen

Sie können die Definition eines virtuellen Feldes mit einem Filter beenden, d.h. Sie können eine Bedingung für die Berechnung des Feldwerts festlegen. Dieser Filter entspricht einschränkenden Bedingungen in der Abfrage (wie in MS-Query).

Beispiel:
Sie wollen die Summe der Felder "res1" für den Fall erhalten, daß "res2" größer als ein bestimmter Wert ist. Diese Bedingung für "res2" ist ein Filter.

Tun DB Revamp erlaubt Ihnen, virtuellen Feldern einen Filter zuzuweisen. Der Filter wird verwendet, sobald der Anwender das Feld nutzt. Es gibt folgende Filter:

- Statisch: fester Filterwert.
- Dynamisch: die Anwender geben bei einer Anfrage ihre eigenen Werte ein.

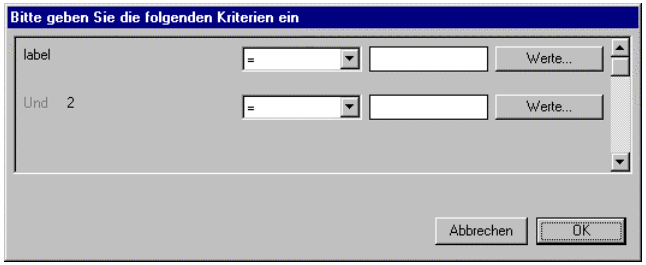
Um einem virtuellen Feld einen Filter zuzuweisen, markieren Sie das Feld und klicken Sie auf die Registerkarte **Filter** in der **Eigenschaftenbox**. Danach:

- Geben Sie eine Bezeichnung für den Filter in das entsprechende Feld ein. Für einen statischen Filter ist die Bezeichnung optional. Für einen dynamischen Filter muß die Bezeichnung des Filters, für den der Anwender einen Wert eingeben muß, den Zweck des Filters andeuten.
- Wählen Sie aus dem Listenfeld **Feld** die Tabelle und das Feld, auf das der Filter anzuwenden ist.
- Wählen Sie aus dem Listenfeld **Vergl** den Vergleichsoperator.
- Für statische Filter geben Sie den Filterwert in das Feld **Wert** ein. Für dynamische Filter geben Sie ein Fragezeichen **(?)** ein.
- Klicken Sie auf **Einf**, um das definierte Kriterium einzufügen.

Sie können einen Satz von Bedingungen bzw. Kriterien erstellen. Definieren Sie die Kriterien wie oben beschrieben, wählen Sie **Und** oder **Oder**, um das neue Kriterium hinzuzufügen.

Um zu überprüfen, ob die den erstellten Felder zugewiesenen Rechenoperationen Ihren Wünschen entsprechen, verwenden Sie **Tun Dialogbox Revamp's** Abfragefunktion. Siehe "**Reale und wirkliche Datenbanken abfragen**".

Ist der Filter dynamisch, erscheint bei Abfragen virtueller Felder ein Fenster ähnlich dem folgenden:



Geben Sie die entsprechenden Werte ein, um den Filter für das virtuelle Feld anzuwenden. Klicken Sie auf die Schaltfläche **Werte...**, um eine Liste für die möglichen Werte für das aktuelle Feld anzuzeigen.

Tabellenübergreifende Verknüpfungen

Die in einer virtuellen Tabelle definierten Felder werden aus einer oder mehreren Tabellen einer wirklichen Datenbank erlangt.

Für jede virtuelle Tabelle kommt es darauf an, die Verknüpfungen zwischen den wirklichen Tabellen, aus denen die Bestandteilsfelder stammen, zu definieren. Die Definition dieser Verknüpfungen ist die Voraussetzung zur Erzeugung von Verknüpfungen zwischen den wirklichen Tabellen, wenn der Endanwender eine virtuelle Datenbank abfragt. Diese Verknüpfungen können direkt oder indirekt sein (d. h. Verknüpfungen zwischen einer und einer anderen Tabelle oder mehr als einer Tabelle und anderen Tabellen).

➢ Definition von Verknüpfungen

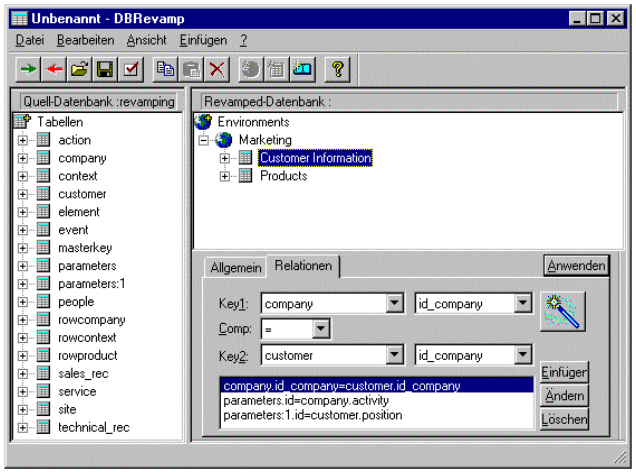
Der einfachste Weg ist die gleichzeitige Definition von Verknüpfungen bei der Erzeugung von Feldern in einer virtuellen Tabelle. Alle wirklichen Tabelle, die für die Felddefinition gebraucht werden, müssen direkt oder indirekt mit den anderen wirklichen Tabellen, die von der virtuellen Tablle benutzt werden, verknüpft sein.

Um Verknüpfungen zwischen realen Tabellen für dieselbe virtuelle Tabelle zu definieren, wählen Sie die virtuelle Tabelle und gehen wie folgt vor:

- Klicken Sie das **Relationen**-Register der virtuellen Tabelle.
- Für jede involvierte reale Tabelle wählen Sie den realen Tabellennamen und das Feld, das für die Verbindung (join) mit der anderen Tabelle verwendet werden soll. Dazu benutzen Sie Sie die Listboxen **Key1** für die erste reale Tabelle und **Key2** für die zweite reale Tabelle.

Hinweis:
Die Namen der zwei Felder, die eine Verknüpfung herstellen, können verschieden sein, auch wenn Sie die gleichen Daten enthalten.

- Auswahl eines vergleichenden Operator in der **Comp** Listbox.
- Klicken des Button **Einfügen**, um die Verknüpfung der Verknüpfungsliste in der virtuellen Tabelle hinzuzufügen.



Sie können eine Verknüpfung mit dem Button **Ändern** nach Änderung der Werte im entsprechenden Element modifizieren. Klicken Sie den **Entf** Button, um das ausgewählte Element zu löschen.

Klicken Sie den **Anwenden** Button, um die so definierte Verknüpfungsliste gültig zu machen.

➢ Überprüfung von Verknüpfungen

Tun DB Revamp stellt eine Funktion bereit, die zur Überprüfung der vom Administrator für die Definition der virtuellen Tabelle erstellten Verknüpfungen zwischen den wirklichen Tabellen dient.

Sie können leicht bei jeder virtuellen Tabelle prüfen, ob die wirklichen Tabellen, die benutzt werden, verknüpft sind und die Verknüpfungen ein in sich kohärentes Ganzes bilden.

Um das zu tun, klicken Sie auf den Button  in der **Relationen** Tabelle.

Tun DB Revamp wird dann alle vom Administrator erstellten Verknüpfungen untersuchen und möglicherweise fehlende direkte oder indirekte Verknüpfungen, die bestimmte Tabellen vom Rest isolieren, entdecken.

Fehlt eine Verknüpfung zwischen zwei Tabellen, versucht **Tun DB Revamp** sie über zwei namensgleiche Felder zu verknüpfen.

Existieren die beiden Felder, schlägt **Tun DB Revamp** die Verknüpfung folgendermaßen vor:

In den meisten Fällen wird die vorgeschlagene Verknüpfung die richtige sein. Sollte jedoch die Verknüpfung Ihrer Meinung nach nicht die richtige sein, definieren Sie die Verknüpfung manuell wie in "**Uberprüfung von Verknüpfungen**" beschrieben.

Wenn auf der anderen Seite zwei unverknüpfte Tabellen keine namensgleichen Felder besitzen, gibt **Tun DB Revamp** eine Liste der unverknüpften Tabellen aus:

In diesem Fall definieren Sie die Verknüpfung(en) manuell, wie im Abschnitt "**Uberprüfung von Verknüpfungen**" beschrieben.

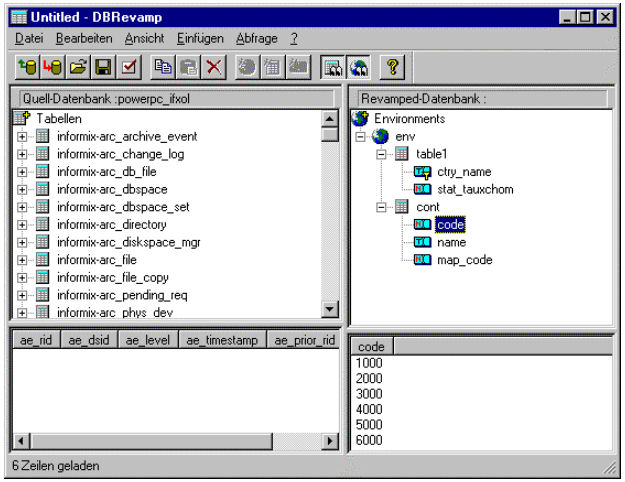
Reale und virtuelle Datenbanken abfragen

Tun DB Revamp enthält eine Abfragefunktion für Tabellen und Felder realer und virtueller Datenbanken.

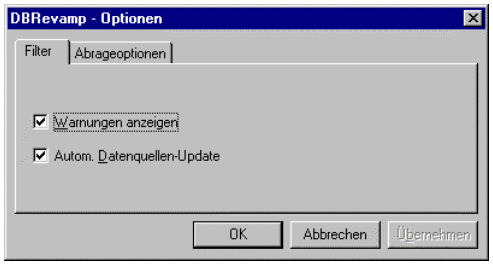
Mit dieser Funktion können Sie eine Tabelle oder ein Feld einer realen oder virtuellen Datenbank direkt aus Tun DB Revamp betrachten, ohne ein Abfragetool wie MS-Query zu verwenden.

Um eine Tabelle bzw. ein Feld einer realen Datenbank abzufragen, wählen Sie vom Hauptmenü Abfrage➔Quelle (oder Abfrage➔Environment) oder klicken Sie in der Werkzeugleiste auf  .

Je nach der gewählten Option erscheint ein Bereich unter der realen oder virtuellen Datenbank. Sie können beide Optionen gleichzeitig wählen.



Um bei einer Tabellen- oder Feldabfrage die Anzahl der Datensätze und die Spaltenbreite einzuschränken, wählen Sie aus dem Hauptmenü **Ansicht➔Optionen...** und klicken Sie auf die Registerkarte **Abfrageoptionen**:

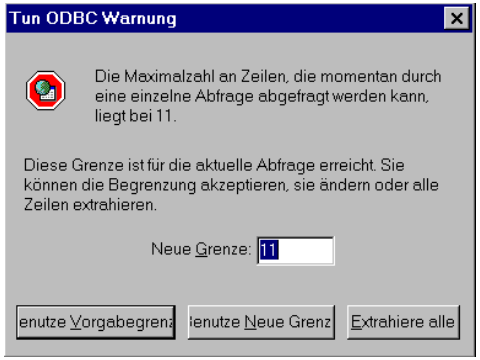


Geben Sie die maximale Anzahl der anzuzeigenden Datensätze und die maximale Spaltenbreite in die entsprechenden Felder ein.

Hinweis:
Einschränkungswerte, die beim Erstellen der Datenquelle definiert wurden (siehe "Datenquelle erstellen") haben eine höhere Priorität als die unter **Abfrageoptionen** eingegebenen Werte.

Beispiel:
Beim Erstellen einer realen Datenqulle wurde eine variable Obergrenze von 11 bis 15 Zeilen definiert.

Fragen Sie eine reale Datenbank mit mehr als 11 Datensätzen ab, erscheint eine Meldung ähnlich der folgenden:



Der genaue Text der Warnung hängt von der Art der Begrenzung ab.

Eine Umgebung gültig machen

Tun DB Revamp beinhaltet eine Funktion, die die Konsistenz zwischen den Inhalten der durch den Administrator erstellten Umgebungen und denen der wirklichen Datenbanken prüft. Dies ist vor allem dann nützlich, wenn an der wirklichen Datenbank Veränderungen vorgenommen wurden (z. B. Ändern oder Löschen eines Feldes), die vom Administrator noch nicht in die virtuelle Datenbank übertragen wurden.

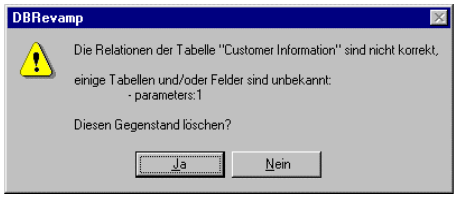
Um diese Funktion zu nutzen, muß die Umgebung auf ihre Gültigkeit geprüft werden, bevor sie exportiert wird, um mögliche Inkonsistenzen zu vermeiden. Benutzen Sie die **Datei→Environments prüfen...** im Hauptmenü, oder klicken Sie den Button  in der Werkzeugleiste.

Zum Beispiel: Tabelle "parameters:1" und ihre Felder wurden in die Umgebung "Marketing" kopiert. Jedes auf diese Art entstandene Feld in der virtuellen Tabelle hat als Ursprungsort die Tabelle "parameters:1".

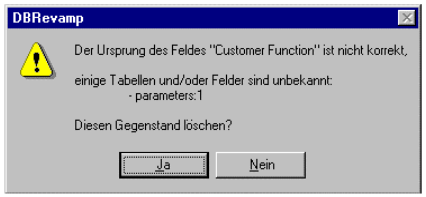
Wenn diese logische Tabelle in der wirklichen Datenbank in der Folge gelöscht wird, haben die Felder der virtuellen Tabelle keinen Ursprungsort mehr. Gleichermaßen werden alle Tabellen, die Referenzen auf die Tabelle "parameters:1" enthalten, in ihren Beziehungen inkonsistent.

Auf Anfrage um Umgebungsvalidierung tut **Tun DB Revamp** folgendes:

- Zeigt die virtuellen Tabellen an, die eine oder mehrere Relationen mit der gelöschten Tabelle haben und bietet deren Löschung an. In diesem Fall empfiehlt es sich, nicht die Tabelle zu löschen, sondern lediglich die Felder, die auf die gelöschte wirkliche Tabelle zurückgehen, zu entfernen.



- Zeigt die virtuellen Felder, deren Ursprung in der gelöschten Tabelle liegen, an und bietet deren Löschung an.



- Wird keine Inkonsistenz festgestellt, erscheint das folgende Fenster:

Umgebungen von Datenquellen exportieren

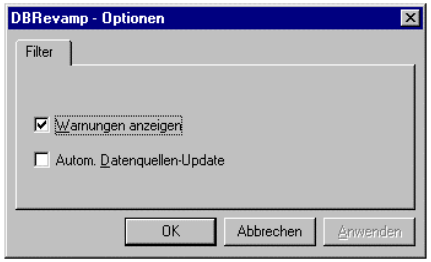
Um eine durch ihre Umgebung neu definierte Datenbank den Anwendern zugänglich zu machen, müssen Sie sie vom PC zum Server exportieren.

Dazu wählen Sie im Hauptmenü **Datei→Exportieren...** oder klicken sie auf  .

Dieser Vorgang generiert oder aktualisiert die drei Tabellen mit der Information über die überarbeiteten Datenbanken. Siehe das Kapitel "**Revamping in Tun SQL**".

Aktualisierung einer virtuellen Datenquelle

Wenn Sie den Namen einer Umgebung ändern, können Sie die korrespondierende virtuelle Datenquelle aktualisieren. Diese Funktionalität kann automatisch gegeben sein, wenn Sie **Automatische Datenquellen-Update** in der Checkbox **Ansicht→Optionen** gewählt haben.



Wurde diese Option gewählt, gibt es zwei Möglichkeiten:

- Wenn die Checkbox **Warnungen Anzeigen** angewählt ist, wird **Tun DB Revamp** um Bestätigung nachfragen, wann immer die automatische Aktualisierung angestoßen wird.
- Ist die Checkbox nicht angewählt, erfolgt die automatische Aktualisierung ohne Bestätigung.

Auf der anderen Seite können Sie, wenn die Option **Automatische Datenquellen-Update** nicht aktiviert ist, die Hauptmenüoption **Einfügen→Update Datenquelle** oder die Kontextmenüoption **Update Datenquelle** für die fragliche Umgebung anwenden, um die entsprechende Datenquelle zu aktualisieren.

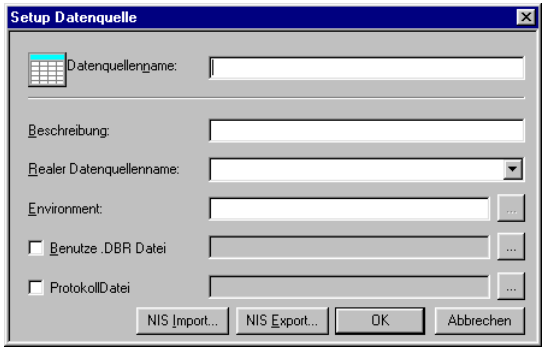
Wollen Sie zeitweise die Verknüpfung zwischen einer Umgebung und ihrer Datenquelle aufheben, benutzen Sie die Option **Einfügen→Löschen Datenquelle** oder die Kontextmenüoption **Löschen Datenquelle**. Die Verknüpfng kann später über die Option **Update Datenquelle** wieder aufgebaut werden.

Erzeugung einer virtuellen Datenquelle

Das Kapitel "**Konfiguration und Nutzung**" beschreibt, wie eine virtuelle Datenquelle mit Hilfe des ODBC Administrators erzeugt wird.

Sie können dazu auch mit **Tun DB Revamp** arbeiten. Wählen Sie **Erzeuge verbundene Datenquelle...** aus dem Kontextmenü des Environments.

Die folgende Dialogbox erscheint:



Geben Sie eine Beschreibung der Datenquelle in das Feld **Beschreibung** ein. Markieren Sie die Checkbox **Protokolldatei** und spezifizieren Sie diese, falls Sie ein Activity-Trace für die virtuelle Datenquelle vorhalten wollen. Weitere Informationen zu dieser Dialogbox finden Sie im Abschnitt "**Erzeugung einer virtuellen Datenquelle**" im Kapitel "**Konfiguration und Nutzung**".

Anzeige von Warnungen

Eine bestimmte Anzahl von Warnungen kann dem Administrator bei der Nutzung von **Tun DB Revamp** angezeigt werden. Diese Warnungen enthalten Informationen beispielsweise über Inkonsistenzen, die während der Umdefinierung einer Datenbank auftreten oder über das Fehlen von Elementen in der neuen Struktur warnen.

Tun DB Revamp zeigt diese Warnungen standardmäßig an. Sie können jedoch die Ausgabe von Messageboxen unterdrücken, indem Sie die Checkbox **Warnungen anzeigen** (**Ansicht→Optionen**) deaktivieren.

Lokales Management von überarbeiteten Datenquellen

Sie können die Beschreibung der revampten Datenbank lokal ändern und speichern. Dies kann sinnvoll sein, wenn Sie die revampte Datenbank nicht sofort exportieren wollen oder frühere Versionen beibehalten wollen. Diese Beschreibung wird in einer Datei mit der Erweiterung "**.dbr**" gespeichert.

➢ Lokal speichern

Um lokal eine mit **Tun DB Revamp** erstellte Beschreibung einer virtuellen Datenbank zu speichern, benutzen Sie die Option **Datei→Speichern** (oder **Datei→Speichern unter...**, um sie unter einem anderen Namen zu sichern) oder ansonsten durch Klicken auf den Button  in der Werkzeugleiste.

Der Pfad für die wirkliche Datenquelle wird ebenso gespeichert, wodurch die virtuelle Datenquelle später exportiert werden kann, ohne deren Herkunft bekanntzumachen.

➢ Öffnen einer lokalen Datenquelle

Um eine virtuelle Datenquelle, die lokal in einer "**.dbr**" Datei gespeichert ist zu öffnen, wählen Sie **Datei➔Öffnen** aus dem Hauptmenü oder klicken den Button  in der Werkzeugleiste.

Wählen Sie die gewünschte revampte Datenquelle (eine "**.dbr**" Datei).

Die zuletzt benutzten Datenquellen können Sie direkt aus dem **Datei**-Menü heraus öffnen.

➢ Datenbankstruktur neu laden

Öffnen Sie eine lokal gespeicherte "**.dbr**"-Datei,, so können Sie die Datenbankstruktur aktualisieren, aus der die Umgebungen erstellt wurden. Das ist hilfreich, wenn die reale Datenbank seit dem letzten Speichern der "**.dbr**"-Datei geändert wurde. Dazu wählen Sie im Hauptmenü **Datei➔Datenbankstruktur neu laden.**

Nach diesen Vorgang empfehlen wir, daß Sie die zuvor generierten Umgebungen validieren, insbesondere, falls die zum Definieren der virtuellen verwendeten realen Felder verschoben oder gelöscht wurden. Siehe "Validieren einer Umgebung" für weitere Einzelheiten.

Feldidentifikation

➢ Feldsymbole

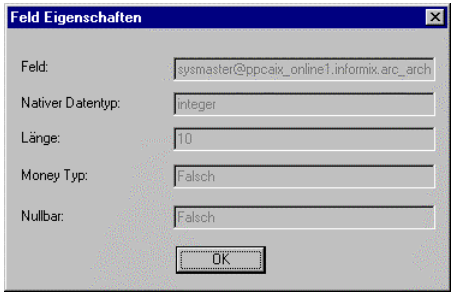
Tun DB Revamp verwendet Feldsymbole, um das Lesen realer oder virtueller Tabellen zu vereinfachen:

Symbol	Bedeutung
	Zeichenfeld
	Datumsfeld
	Numerisches Feld
	Binarfeld
	Primärer Tabellenschlüssel

➢ Eigenschaften realer Felder

Wollen Sie weitere Information über ein Feld, können Sie vom Kontextmenü eines Felds einer realen Tabelle **Eigenschaften** (linke Fläche) wählen.

Es erscheint die folgende Dialogbox:



Hinweis:
Der hier gezeigte Feldtyp entspricht dem systemeigenen Typ und hängt daher von der verwendeten DBMS ab.

TEIL 3
ANHÄNGE

REFERENZEN

Index

Hinweis:
xxx steht für die relative Dateierweiterung einer bestimmten Datenbank. Die Dateierweiterungen sehen folgendermaßen aus:

- **ifx** Informix
- **ora** Oracle
- **syb** Sybase
- **db2** DB2
- **pro** Progress
- **pro7** Progress7
- **pro8** Progress8
- **ism** C-ISAM
- **mvs** DB2 für MVS

CONFIG.xxx	Datei mit Arbeits- und Sicherheitsparametern des **Tun SQL** UNIX Server
DBMAP	Windows Applikation zur Anlage oder Änderung von Zeichenkonvertierungstabellen
DBSCRIPT	Windows Applikation, die SQL Batchdateien interpretiert und ausführt
DBSHOW	Windows Applikation für Test und Konfiguration
PARAM.xxx	Datei mit Einstellungsparametern des **Tun SQL** UNIX Server
TUNODBC200.xxx	**Tun SQL** UNIX Server

Datei mit Arbeits und Sicherheitsparametern des **Tun SQL** UNIX Server

➢ Beschreibung

Die **config.xxx** Dateien kann eine bestimmte Anzahl von Parametern für den **Tun SQL** UNIX Server enthalten. ImGegensatz zu den **param.xxx** Dateien betreffen die Parameter nicht den grundsätzlichen Betrieb des Servers, sondern die eine oder andere Datenbank. Zum Beispiel kann eine **config** Datei so aussehen:

```
#Optional declaration for databases
#Example :
#[base_Name]
#Define=ENV_VARIABLE:value
#RowLimitMode=None|Absolute|Fixed|Variable|Extended|1
|2|3|4|5
#RowLimitValue=value
#RowLimitMax=value
#DbmsName=DatenbaseName
#Version=DatenbaseVersion

# In this section, list allowed configuration
(base,user,Product)
# Base_Name|*,User_Name|*,Product_Name|*
[Allowed]

# In this section, list denied configuration
(base,user,Product)
# Base_Name|*,User_Name|*,Product_Name|*
[Denied]
```

Diese Datei hat ebensoviele Abschnitte wie von dem DBMS verwaltete Datenbanken, die mit dem **Tun SQL** UNIX Server verbunden sind. Es ist nicht notwendig alle Datenbanken aufzuführen, wenn keine speziellen Parameter zu setzen sind.

Jeder Abschnitt enthält in eckigen Klammern den Namen der Datenbank (zum Beispiel, [**tunsqldemo**]). Die folgenden Parameter können in jedem Abschnitt definiert werden:

Define=ENV_VARIABLE:value

Weist einen Wert der Umgebungsvariable **ENV_VARIABLE** vor dem Öffnen der Datenbank (Installationsverzeichnis der Datenbank, Datenformat...) zu. Diese Option kann so oft wie nötig in der **config** Datei vorkommen.

#RowLimitMode=None|Absolute|Fixed|Variable|Extended|1|2|3|4|5
#RowLimitValue=value
#RowLimitMax=value

Einige Büroapplikationen geben dem Benutzer die Möglichkeit seine eigenen SQL Requests zu definieren. In anderen Fällen neigen bestimmte Applikationen dazu den gesamten Inhalt einer Tabelle zu lesen, bevor sie die Daten am Bildschirm darstellen. Dies ist kein Problem, solange es sich um kleine lokale Datenmengen handelt. Sind jedoch riesige Tabellen einer zentralen Datenbank betroffen, kann das zu unüberwindbaren Problemen führen. In diesem Fall wächst der Netzwerkverkehr enorm an und der Speicher des PCs läuft über. Der PC muß häufig neu gestartet werden nach derartigen "select" Requests. Um dieses Problem zu kompensieren, enthält der **Tun SQL** ODBC Treiber die Begrenzungsoptionen, die durch den Parameter **RowLimitMode** gesetzt werden können. Wenn dieser Parameter gesetzt ist, hat er Vorrang gegenüber dem möglicherweise für die Datenquelle definierten Wert am PC.

None	Keine Grenze wird durch den ODBC Treiber gesetzt
Absolute	Der ODBC Treiber wird nicht mehr als **RowLimitValue** Zeilen während eines **select** Requests laden. Der Anwender erhält keine Nachricht.
Fixed	Der ODBC Treiber wird nicht mehr als **RowLimitValue** Zeilen während eines **select** Requests laden. Der Anwender erhält eine Bildschirmmeldung.
Variable	Der ODBC Treiber wird nicht mehr als **RowLimitValue** Zeilen während eines **select** Requests laden. Der Anwender erhält eine Bildschirmmeldung, die ihm das Laden von mehr innerhalb der Grenzen des **Maximalwertes** (**RowLimitMAX)** nahelegt.

Extended	Der ODBC Treiber wird nicht mehr als **RowLimitValue** Zeilen während eines **select** Requests laden. Der Anwender erhält eine Bildschirmmeldung, die ihm das Laden von mehr ohne einen Maximalwert anzeigt. In diesem Fall ist die Meldung lediglich eine Warnung.

DbmsName=DatabaseName

Dieser Parameter dient dem Setzen bzw. Ändern des Namen der Datenbank, der dem ODBC Treiber mitgeteilt wird.

Version=DatabaseVersion

Dieser Parameter dient dem Setzen oder Ändern der Versionsnummer der Datenbank, die dem ODBC Treiber mitgeteilt wird.
Zusätzlich zu den den einzelnen Datenbanken zugeordneten Abschnitten kann die config-Datei die Abschnitte **[Allowed]** und **[Denied]** die zum Zugriffsschutz auf bestimmte Datenbanken benutzt werden können. Beide Funktionen arbeiten folgendermaßen:

[Allowed]

Dieser Abschnitt muß Folgen von drei Parametern enthalten, wie z.B.:

```
base_name,user_name,product_name
```

wobei

- **base_name** der Name einer Dataenbank ist
- **user_name** der Name des Benutzer einer Dataenbank ist
- **product_name** der Name einer Windows Applikation ist, die den Server benutzen wird.

Jede Dreierfolge zeigt an, daß ein Benutzer (**user_name**) *berechtigt* ist, die Datenbank (**base_name**) mit der Windows Applikation (**product_name**) zu benutzen. Jeder Parameter kann durch das allgemeingültige Zeichen * ersetzt werden.

Zum Beispiel bedeutet die Dreierfolge **tunsqldemo,*, excel**, daß alle Benutzer die Datenbank **tunsqldemo** mit der Applikation **excel** benutzen dürfen; ausgenommen davon sind nur jene Namen, die in einer ähnlichen Dreierfolge im Abschnitt **[Denied]** enthalten sind.

[Denied]

Dieser Abschnitt sollte Dreierfolgen enthalten, wie sie für den Abschnitt **[Allowed]** definert sind.

Jede Dreierfolge zeigt an, daß ein Benutzer (**user_name**) *nicht berechtigt* ist, die Datenbank (**base_name**) mit der Windows Applikation (**product_name**) zu benutzen. Jeder Parameter kann durch das allgemeingültige Zeichen * ersetzt werden. So bedeutet die Dreierfolge ***,john,excel**, daß der Benutzer **john** keine einzige Datenbank mit der Applikation **excel benutzen** darf. Ausnahmen davon müssen in einr entsprechenden Folge im Abschnitt **[Allowed]** definiert werden.

Hinweis:
Damit ein **Tun SQL** UNIX Server die **config**-Datei berücksichtigt, muß sie bei der Befehlseingabe mit der Option -c dem Server bekanntgemacht werden.

Es können zwei verschiedene Informix Datenbank Engines koexistieren (Informix Version 5 und Informix Version 7). Beispielsweise könnte eine Datenbank über Datenbank Engine 1 aus dem Verzeichnis /u/informix1 und eine zweite von der Datenbank Engine 2 aus dem Verzeichnis /u/informix2 zugreifbar sein. In diesem Fall enthält die Datei **config.ifx**:

```
[database1]
Define=INFORMIXDIR:/u/informix1
Define=DBPATH:/u/database1
Version=5.01
[database2]
Define=INFORMIXDIR:/u/informix2
Define=DBPATH:/u/database2
Version=7.01
```

➢ Siehe auch

param.xxx, tunodbc200.xxx

Windows Applikation zur Anlage oder Änderung von Konvertierungstabellen.

➢ Syntax

```
DBMAP [-ffile_name]
```

➢ Beschreibung

Tun DB Map kann zum Anlegen oder Ändern von Zeichenübersetzungstabellen. Mit den Übersetzungstabellen können Probleme vermieden werden, die durch die unterschiedliche Kodierung von Sonderzeichen auf den verschiedenen Plattformen hervorgerufen werden kann. Damit die Übersetzungstabellen berücksichtigt werden, müssen sie in der Definition der betreffenden Datenquelle angegeben werden.

-ffile_name

Parameter für die Angabe einer vorhandenen Übersetzungstabelle.

Windows Applikation zur Ausführung von SQL Batchdateien

➢ Syntax

```
DBSCRIPT [-ddata_source] [-ffile_name]
```

➢ Beschreibung

Tun DB Script ermöglicht die Ausführung einer ganzen Liste von SQL Requests in einer einzigen Operation. Es interpretiert die SQL Kommandos nacheinander und stoppt beim Auftreten eines Fehlers. **Tun DB Script** kann auch zum Sichern und Ändern von SQL Batchdateien benutzt werden.

Tun DB Script ist nützlich für das Herunterladen des Inhalts einer Datenbank von enem PC unter Windows auf ein remotes DBMS. Es kann auch für die Aktualisierung und Bereinigung großer Datenbanken eingesetzt werden.

-ffile_name

Wird zur Angabe des Dateinamens der die SQL Kommandos enthaltenden Datei beim Start der Applikation benutzt.

-ddata_source

Wird zur Angabe der Datenquelle benutzt, mit der die SQL Batchdatei arbeiten wird. **Tun DB Script** erzeugt nicht automatisch die Verbindung mit der Datenquelle. Dies muß anderweitig "manuell" erfolgen.

Windows Applikation für Tests und Konfiguration

➢ Syntax

```
DBSHOW [-hhost_Name]
```

➢ Beschreibung

Diese Applikation kann benuzt werden, um einen UNIX Host im Netzwerk auf die Existenz eines oder mehrerer **Tun SQL** UNIX Server hin abzufragen.

Geben Sie den Namen des Hosts im Feld **Hostname** ein, klicken Sie dann auf den Button **Host abfragen** um diese Information zu erhalten. Wenn einer oder mehrere **Tun SQL** UNIX Server auf der remoten Maschine korrekt installiert sind, nennt **Tun DB Show** deren Namen und die der damit verbundenen DBMSs.

Dieses Programm ist besonders für die Überprüfung der Installationsgüte geeignet.

-hhost_Name

Gibt den Namen des abzufragenden Hosts an.

Datei mit den Laufzeitparametern des **Tun SQL** UNIX Servers.

➢ Beschreibung

Statt eine Vielzahl von Befehlszeilenparamter dem **Tun SQL** UNIX Server beim Start zu übergeben, empfiehlt es sich alle diese Optionen in eine Datei zu schreiben und den Namen dieser Datei dem Host mit der Option **-f** mitzuteilen.

Die **Tun SQL** Installationsprozedur nutzt diesen Mechanismus und schreibt diese Optionen in **param.xxx** Dateien, wobei das Zeichen ***** durch die Abkürzung für das entsprechende DBMS (**param.ora**, **param.syb**, **param.ifx**) ersetzt wird.

Hier ist ein Beispiel für eine solche Datei:

```
-output=/dev/null
-output2=/dev/null
-DORACLE_HOME=/home3/oracle/7.1.4
-DORACLE_SID=odbc
-config=/usr/tunsql/config.ora
```

Die Bedetung der einzelnen Optionen ist im Abschnitt **tunodbc200.xxx** erklärt.

➢ Progress

In manchen Progress-Feldern können verschiedene Werte aufgeführt sein. Sie sind unter dem Namen "Array Fields" bekannt. Um diese Werte in Anwendungen, wie beispielsweise MS Query und MS Access, anzuzeigen, muß in der Datei param.proX (wobei X die Nummer der verwendeten Progress-Version ist) folgende Option angegeben sein:

```
-arrayfields=*
```

wobei * ein Platzhalter für eins der folgenden Zeichen ist:

$, &, #, %, - ,_.

Das Standardzeichen ist _.

Die zur Anzeige der verschiedenen Array-Feld-Werte erforderlichen Spalten erhalten anschließend einen Namen:
Spaltenname*n*,
wobei * das in der Datei param.proX (standardmäßig "_") gewählte Zeichen ist, und n die Position des Werts in der Tabelle darstellt.

Beispiel:
Der zweite Wert in dem Array-Feld wird in Spalte col_2_ angezeigt.

Sollte die Verwendung des Zeichens "_" ein Problem verursachen, wenn die Spalte generiert wird (andere Spalten tragen möglicherweise einen ähnlichen Namen), müssen Sie eins der anderen vier Zeichen wählen.

➢ Siehe auch

config.xxx, tunodbc200.xxx

Tun SQL UNIX Server.

Hinweis:
Die verschiedenen **Tun SQL** UNIX Server haben eine bestimmte Anzahl gemeinsamer Optionen. Die Liste unterschiedlicher Optionen kann durch Aufruf der ausführbaren **tunodbc.xxx** mit der Option **-a[ll]** angezeigt werden, wobei xxx hier die relative Erweiterung der fraglichen Datenbank darstellt.

➢ Syntax

```
tunodbc200.XXX
  -a[ll]
  -c[onfig]=config_file
  -Dname=value
  -db[ms]=DBMS_name
  -de[bug]
  -f[ile]=param_file
  -h[old]
  -i[nter]
  -l[owercase]
  -n[opassword]
  -nor[owcount]
  -o[utput]=file_name
  -o[utput]2=file_name
  -ow[ner]
  -p[rogress]=XX
  -s[electby]
  -sv[archar]
  -sy[scolumns]
  -t[imer]=xx
  -u=user1,user2...
  -v[ersion]=DBMS_version_number
  -x=user1,user2...
```

➢ Beschreibung

-a

Listet alle von **tunodbc200.xxx** unterstützten Optionen.

-c=config_Datei

Verbindet eine Konfigurationsdatei (**config.xxx**) mit dem Server (Cf. config.xxx).

-db=Name

Wird benutzt, um einen DBMS Namen mit dem **Tun SQL** UNIX Server zu verbinden. Dies ist der Wert, der in der Liste beim Aufruf von **Tun DB Show** erscheint.

-de

Gibt dem Server den Protokollmodus (trace mode) mit. Die Meldungen erscheinen standardmäßig auf **/dev/console**.

-DName=value

Setzt die Umgebungsvariable "**name**" auf den Wert "**value**" vor Ausführung des Servers (Installationsverzeichnis der Datenbank, Datenformat...). Diese Option kann so oft wie nötig in der Befehlszeile wiederholt werden. Es ist absolut unerläßlich, einige dieser Variablen zu definieren, um mit einem **Tun SQL** UNIX Server mit bestimmten DBMSs arbeiten zu können. Die Definition wird durch die Installationsprozedur vorgenommen.

Aus Referenzierungsgründen werden diese Variablen für folgende DBMSs benötigt:

Oracle:

ORACLE_HOME: Oracle Installationsverzeichnis
ORACLE_SID: Standard Datenbank

Informix

INFORMIX_DIR: Informix Installationsverzeichnis
DBPATH: Verzeichnis für die Datenbank (nur SE)

Sybase

SYBASE: Sybase Installationsverzeichnis
SYBSERVNAME: Benennung der Server-Setup-Datei (Optional). Dieser Wert ist identisch mit der von SYBASE definierten und benutzten Variable DSQUERY.

-f=param_file

Gibt den Namen einer Datei an, in die alle für das Programm definierten Optionen nacheinander aufgeführt sind. Diese Option kann dem Programm mitgegeben werden, so daß nicht Dutzende von Optionen benannt werden müssen.

-i

Benutzt den interaktiven Testmodus zur Feststellung des ordnungsgemäßen Betriebs des Servers.

-o=file

Zeigt den Namen der Datei oder des Gerätes an, in das die Server Protokollnachrichten und ihr Zugriffskontrollmechnismus (Watchdog) geschrieben werden. Arbeitet nur im **Debug** Modus.

-o2=file

Gibt den Namen der Datei oder des Gerätes an, auf die die Protokollmeldungen des Watchdog ausgegeben werden.. Arbeitet nur im **debug** Modus.

-t=xx

Benennt einen max. Wartezeit-Wert. Dies ist die Zeit, nach der die vollkommene Abwesenheit einer Antwort durch den PC als Ausfall gewertret wird. Der **Tun SQL** UNIX Server wird daher seine Arbeit ebenso einstellen. Dieser Wert ist nicht so wichtig wie der, der auf dem PC bei der Definition der Datenquelle definiert wird.

-u

Wird mit Parameter verwendet, um die Liste berechtigter Benutzer (authorized users) zu definieren ("*****" für everyone). Default ist *****. Beispiel:

```
-x = *
-u = bill     (nur "bill" ist autorisiert)
```

-v=XX

Verbindet eine DBMS Versionsnummer mit dem **Tun SQL** UNIX Server. Es ist ein Wert, der beim Aufruf von **Tun DB Show** mit angezeigt wird.

-x

Für die Definition nicht-zugriffsberechtigter Benutzer. ("*****" für everyone). Beispiel:

```
-u = *
-x = bill
```
(nur "bill" ist nicht autorisiert)

➢ Informix Optionen

-h

Standardmäßig hält Informix die Cursor während der Ausführung der Kommandos **commit** und **rollback** nicht offen. Die Option -h stellt sicher, daß die Cursor bei einem solchen Kommando offen bleiben, wenn die Applikationen, die diesen Servertypen benutzen, dazu nicht in der Lage sind.

-n

Gilt nur für SCO UNIX 5. Wenn das System SCO UNIX 3.2 Version 5 Benutzerpasswörter nicht korrekt prüfen kann, setzt diese Option die Passwortprüfung aus.

-s

Standardmäßig kann Informix keine **select** Kommandos mit einer Sortieroption (**group by** oder **order by**) in einer Spalte, die nicht im **select** Kommando erwähnt wird, wahrnehmen. Die Option **-s** kann diesen Mangel kompensieren, wenn Applikationen das vorsehen.

➢ Oracle Optionen

-l

Falls Tabellen, Spalten, Indizes oder Ansichten (views) im Katalog mit Kleinbuchstaben erzeugt wurden, stellt die Option -l sicher, daß die Katalogfunktionen in Anführungszeichen eingeschlossene Daten zurückgeben. Applikationen, die diesen Server benutzen, werden Abfragen (queries) erzeugen, die Namen in Anführungszeichen enthalten.

➢ Progress Optionen

-n

Gilt nur für SCO UNIX 5. Wenn das System SCO UNIX 3.2 Version 5 Benutzerpasswörter nicht korrekt prüfen kann, setzt diese Option die Passwortprüfung aus.

-nor

Die Progress Datenbank kann keine Auskunft über die Anzahl modifizierter oder gelöschter Zeilen erteilen, wenn ein **update** oder **delete** Kommando ausgeführt wird. Standardmäßig kompensiert dieser Server den erwähnten Mangel. Damit geht jedoch ein Zeitverlust bei jedem **update** oder **delete** Kommando einher. Falls den Server benutzende Applikationen keine Information über die Zahl geänderter Zeilen benötigen, stellt die Option **-nor** die Serverkompensation ab und spart so Zeit.

-ow

Standardmäßig unterstützt der Server nicht die Anzeige von Objekteigentümern in der Datenbank. Tatsächlich versuchen einige Applikationen ein Eigentümerpräfix hinzuzufügen, wenn diese mit Objekten zurückgegeben werden und rufen damit Fehler während der Ausführung hervor. Diese Option, **Eigentümer Hinzufügen**, ist dann sinnvoll, wenn eine Applikation den Objekteigentümer benötigt und die Eigentümer korrekt behandelt.

-p=XX

Falls Progress-spezifische Optionen benutzt werden müssen (beispielsweise die Option -Q die die Datenbank zur Einhaltung der ANSI Norm zwingt), sollte -p=XX gesetzt werden, wobei **XX** durch eine in Anführungszeichen eingeschlossene Option ersetzt wird.

-sv

Die Progress Datenbank kennt nur einen Typen von Zeichenfolgen. Standardmäßig wird bei allen Zeichenfolgen eine feste Länge (SQL_CHAR in ODBC) vorausgesetzt. Falls diese Option benutzt wird, werden die Zeichenfolgen behandelt als hätten sie eine variable Länge (**SQL_VARCHAR** in ODBC).

-sy

Progress kann System- oder verborgene Spalten definieren, die nicht zurückgegeben werden, wenn ein Suchbefehl über alle Spalten mittels einer Wildcard abgesetzt wird. Daher beschreibt der Server standardmäßig diese Spalten nicht in seinem Katalog. Falls die verborgenen Spalten jedoch benötigt werden, ruft diese Option deren Ausgabe hervor.

➢ Siehe auch

param.xxx, config.xxx

B

SQL ANWEISUNGEN IN C-ISAM

Die wichtigsten Anweisungen

Syntax der SQL-Anweisung

Die SQL-Anweisungen verwenden die folgende Syntax:

- Reservierte Namen werden in Großbuchstaben eingegeben (INSERT, UNIQUE, etc.). In der Befehlszeile können Sie sie jedoch in Kleinbuchstaben eingeben.
- Namen von Variablen werden kursiv eingegeben (*Databasename, etc.*).
- Eckige Klammern zeigen optionale Parameter oder Eingaben an ([optional]).
- Geschwungene Klammern und der Ausdruck **xor** zeigen eine exklusive Option an ({A xor B xor C}).
- Der Exponent n (^{n}) zeigt eine Sequenz an, die von 0 bis zu n Mal wiederholt werden kann (Sequenz^{n}).

Interpunktionszeichen und Klammern sind Symbole, die genau so eingegeben werden müssen, wie sie aufscheinen.

CREATE DATABASE

➢ Zweck

Erstellen einer neuen Datenbank.

➢ Syntax

CREATE DATABASE *Basename*

Hinweis: Der Name der Datenbank muß kürzer als 18 Zeichen sein.

➢ Anwendung

Die erstellte Datenbank wird die aktuelle Datenbank.

Diese Anweisung kann nur mit **sqltools** (**Tun SQL** tool) verwendet werden.

Nach Erstellung der Datenbank wird ein Verzeichnis namens Databasename.ism erstellt. Dieses Verzeichnis enthält die zusätzlichen C-ISAM-Dateien, aus denen der Katalog besteht (SysTables, SysColumns, SysIndexes, SysDefaults). Es ist ein Unterverzeichnis des aktuellen oder des durch die Umgebungsvariable ISAM-PATH gesetzten Verzeichnisses, falls definiert.

➢ Beispiel

```
CREATE DATABASE TEST;
```

Erstellt das Verzeichnis test.ism mit den Dateien SysTables.dat, SysTables.idx, SysColumns.dat, SysColumns.idx, SysIndexes.dat, SysIndexes.idx, SysDefaults.dat und SysDefaults.idx.

➢ Zweck

Erstellt in der aktuellen Datenbank eine neue Tabelle und unterstellt ihre Spalten oder eine Gruppe von Spalten Datenintegritäts-Einschränkungen.

➢ Syntax

CREATE TABLE *tablename* (Column definition [,Column definition]^{n}) [,Constraint definition] ^{n}

Hinweis: Der Name der Tabelle muß kürzer als 18 Zeichen sein

➢ Anwendung

Die Namen der Tabellen in einer Datenbank müssen eindeutig sein. Jede Spalte in einer Tabelle muß einen anderen Namen haben.

Der Tabellenname kann einen Präfix mit dem Namen des UNIX-Anwenders haben, der der Besitzer der Datenbank wird. Wird kein Name angegeben, so wird die aktuelle Login-ID verwendet.

➢ Beispiel

```
CREATE TABLE TABLE_TEST (c1 char);
```

Erstellt eine Tabelle *table1* in der relationalen Datenbank durch Erstellen der dazugehörigen C-ISAM-Dateien (zum Beispiel *table1_100.dat* und *table1_100.idx*).

Die C-ISAM-Dateinamen werden durch die ersten sieben Zeichen des Tabellennamens und Hinzufügung eines eindeutigen Wertes generiert. Haben Tabellennamen weniger als sieben Zeichen, werden die neuen Dateinamen mit Unterstreichungen (_) aufgefüllt. Der eindeutige Werte ist 100 für die erste Tabelle und erhöht sich mit jeder weiteren Tabelle um 1.

➢ Zweck

Erstellt in der aktuellen Datenbank eine neue Tabelle mit Datenintegritäts-Einschränkungen auf Spalten oder eine Gruppe von Spalten und Bedingungen für die Existenz von Dateien.

➢ Syntax

DEFINE TABLE *tablename* [File is option] (Column definition [,Column definition]^{n}) [,Constraint definition] ^{n}

Hinweis: Der Name der Tabelle muß kürzer als 18 Zeichen sein.

➢ Anwendung

Die Namen der Tabellen in einer Datenbank müssen eindeutig sein. Jede Spalte in einer Tabelle muß einen anderen Namen haben.

Der Tabellenname kann einen Präfix mit dem Namen des UNIX-Anwenders haben, der der Besitzer der Datenbank wird. Wird kein Name angegeben, so wird die aktuelle Login-ID verwendet.

➢ Beispiel

```
DEFINE TABLE TABLE1 file is file_1 (c1 char);
```

In diesem Beispiel bestehen die Dateien *file_1* C-ISAM (*file_1.idx* und *file_1.dat*) bereits; nur die Tabelle muß definiert werden.

Benutzt in den **CREATE TABLE** und **DEFINE TABLE** Anweisungen.

➢ Zweck

Die Option "column definition" in der Anweisung **DEFINE TABLE** (**CREATE TABLE**) listet Namen, Typ, Standardwert und Einschränkung für eine einzelne Spalte.

➢ Syntax

Columnname Data type [Default clause] [Not null clause] [Constraint definition subset]

➢ Beispiel

```
CREATE TABLE PETS
(name char (20),
 race char (25),
 sex  char(1));
```

Diese Tabelle besteht aus den Spalten ***name***, ***race*** und ***sex***.

Benutzt in der **COLUMN DEFINITION** Option.

➢ Syntax

DEFAULT [{Literal xor NULL xor Current xor Today xor User}]

➢ Anwendung

Ist kein Wert angegeben, wird in die Spalte der Standardwert eingefügt. Ist kein Standardwert definiert und die Spalte erlaubt Null, ist der Standardwert NULL.

LITERAL	Eine vom Anwender definierte Zeichenkette oder numerische Konstante.
NULL	Nullwert.
CURRENT	Aktuelles Datum/Zeit (nur mit Typ TIMESTAMP anwendbar).
TODAY	Aktuelles Datum (nur mit Typ DATE anwendbar).
USER	Name des aktuellen Anwenders (nur mit Typ VAR oder VARCHAR anwendbar).

➢ Beispiel

```
CREATE TABLE PETS
(name char (20),
 race char(25),
 sex char(1) DEFAULT 'M')
```

In dieser Tabelle ist der Standardwert für ***sex*** ein Zeichen: 'M'.

Benutzt in der **COLUMN DEFINITION** Option.

➢ Zweck

Geben Sie für eine Spalte keinen Standardwert ein so ist er Null, falls Sie nicht die Schlüsselwörter NOT NULL nach dem Datentyp der Spalte eingeben. In diesem Fall gibt es keinen Standardwert für die Spalte.

➢ Syntax

NOT NULL

➢ Beispiel

```
CREATE TABLE INVOICE
(invoice_id longint NOT NULL,
 customer_name char (30))
```

Ist die Spalte als NOT NULL definiert (und gibt es keinen Standardwert), müssen Sie in diese Spalte einen Wert eingeben, wenn Sie eine Reihe einfügen oder diese Spalte in einer Reihe ändern. Tun Sie das nicht, gibt der Datenbankserver einen Fehler zurück.

Benutzt in der **COLUMN DEFINITION** Option.

➢ Zweck

Mit dem Constraint-Definition-Subset definieren Sie eine Einschränkung für eine einzelne Spalte.

➢ Syntax

{UNIQUE xor PRIMARY KEY} [CONSTRAINT *Constraint name*]

Hinweis: Der Name der Einschränkung muß kürzer als 18 Zeichen und in der Datenbank eindeutig sein.

➢ Anwendung

UNIQUE	Schränkt das Feld auf eindeutige Werte ein.
PRIMARY KEY	Schränkt das Feld auf eindeutige Werte ein und bestimmt es zum Primärschlüssel der Tabelle.

Wird kein Constraintname definiert, wo wird ein Standardwert zugewiesen.

➢ Beispiel

```
CREATE TABLE INVOICE
(invoice_number longint UNIQUE CONSTRAINT un_invoice,
 customer_name char (30))
```

Die Eindeutigkeitseinschränkung der Rechnungsnummer ist ***un_invoice***.

Benutzt in den **CREATE TABLE** und **DEFINE TABLE** Anweisungen.

➢ Zweck

Mit dieser Option können Sie Einschränkungen für eine Gruppe von Spalten (1 bis 8) definieren.

➢ Syntax

{UNIQUE xor PRIMARY KEY} (*Columnname* [,*Columnname*]^{n})[CONSTRAINT *Constraint name*]

➢ Anwendung

UNIQUE	Schränkt das Feld auf eindeutige Werte im Spaltensatz ein.
PRIMARY KEY	Schränkt das Feld auf eindeutige Werte ein und bestimmt es zum Primärschlüssel im Spaltensatz.

Wird für die Einschränkung kein Name definiert, so wird ein Standardname zugewiesen.

Jede in einer Einschränkung angeführte Spalte muß eine Spalte in der Tabelle sein und darf in der Liste der Einschränkungen nicht öfter als ein Mal vorkommen.

➢ Beispiel

```
CREATE TABLE FAMILY
(name char (20),
 surname char (20),
 birth_date date,
 PRIMARY KEY (name, surname) CONSTRAINT pk_family)
```

Die Einschränkung des Primärschlüssels ***pk_family*** gilt für die Felder ***name*** und ***surname***.

Benutzt in dem **DEFINE TABLE** Anweisung.

➢ Zweck

Bestimmt die Anwendung der Tabelle entsprechend der Existenz oder nicht-Existenz einer Datei.

➢ Syntax

FILE IS *filename*

➢ Anwendung

Wird diese Option angewendet, wo werden keine C-ISAM-Dateien generiert. Tun SQL muß die Dateien *filename.dat* und *filename.idx* im Verzeichnis der aktuellen Datenbank finden, bevor es die Tabelle verwenden kann.

Wird die Option nicht verwendet, werden den im aktuellen Datenbankverzeichnis generierten Dateien Standardwerte zugewiesen.

➢ Beispiele

```
CREATE DATABASE TEST;
DEFINE TABLE TABLE1 FILE IS FIC1 (C1 CHAR);
```

In diesem Beispiel werden keine Dateien generiert. Die Dateien *foo1.dat* und *foo1.idx* müssen dem Verzeichnis *test.ism* hinzugefügt werden.

```
DEFINE TABLE TABLE1 (C1 CHAR);
```

Hier werden die Dateien *table_100.dat* und *table_100.idx* (Standardnamen) im Verzeichnis *test.ism* generiert.

➢ Zweck

Erstellt einen Index für eine oder mehrere Spalten in der Tabelle (bis zu acht).

➢ Syntax

CREATE {UNIQUE xor DISTINCT} INDEX *indexname* ON *tablename* (*Columnname* [,*Columnname*] ^{n})

Hinweise: Der Indexname darf nicht länger als 18 Zeichen sein. Die Anzahl der Spalten kann zwischen 1 und 8 sein.

➢ Anwendung

UNIQUE	Schränkt das Feld auf eindeutige Werte ein
DISTINCT	Synonym für UNIQUE

Für Tabellen, die mit der Option FILE IS definiert wurden, muß die Anweisung CREATE INDEX vor dem Kopieren der Dateien filename.dat und filename.idx ausgeführt werden.

➢ Beispiel

```
CREATE DISTINCT INDEX ix_name ON Table1 (name,
birth_date) ;
```

Diese Anweisung erstellt den Index ***ix_name*** für die Spalten ***name*** und ***birth_date*** in der Tabelle ***Table1***.

➢ Zweck

Teilt einer Tabelle ein Synonym zu.

➢ Syntax

CREATE SYNONYM *synonymname* FOR *tablename*

Hinweise: Der Name des Synonyms darf maximal 18 Zeichen enthalten.

➢ Verwendung

Vor den Namen eines Synonyms muß der Name eines UNIX-Benutzers stehen, der dann der Besitzer des Synonyms wird. Sollte kein Name angegeben werden, wird standardmäßig die aktuelle Anmelde-ID verwendet.

➢ Beispiel

```
CREATE SYNONYM angestellter FOR Tabelle1;
```

Mit diese Anweisung wird das Synonym ***angestellter*** für die Tabelle ***Tabelle1*** erstellt.

➢ Zweck

Teilt einer Tabelle oder einem Synonym einen Kommentar zu.

➢ Syntax

COMMENT ON {*tablename* xor *synonymname*} IS 'comment string'

➢ Beispiel

```
COMMENT on TABELLE1 IS 'Angestellter Tabelle'
```

➢ Zweck

Entfernt (löscht) eine gesamte Datenbank, inkl. aller Kataloge, Indizes und der Daten.

➢ Syntax

DROP DATABASE *basename*

Hinweis: Der Name der Datenbank darf nicht länger als 18 Zeichen sein.

➢ Anwendung

Eine von einem anderen Anwender benutzte Datenbank kann nicht gelöscht werden.

Enthält das Datenbankverzeichnis Dateien außer jenen, die für die Tabellen und Indizes der Datenbank erstellt wurden, löscht Anweisung DROP DATABASE die Datenbank nicht.

➢ Beispiel

```
DROP DATABASE DBTEST;
```

Löscht die Datenbank ***DBTEST***.

➢ Zweck

Schließt an eine andere Datenbank an. Sie können eine Datenbank nicht bearbeiten, wenn Sie nicht an sie angeschlossen sind.

➢ Syntax

CONNECT DATABASE *basename*

➢ Beispiel

```
CONNECT DATABASE DBTEST2;
```

Schließt die Datenbank ***DBTEST2*** an.

➢ Zweck

Trennt die zur Zeit angeschlossene Datenbank.

➢ Syntax

DISCONNECT DATABASE *basename*

➢ Beispiel

```
DISCONNECT DATABASE DBTEST2;
```

Trennt die Datenbank ***DBTEST2***.

➢ Zweck

Löscht einen Index.

➢ Syntax

DROP INDEX *indexname*

➢ Beispiel

```
DROP INDEX ix_name ;
```

Löscht den Index ***ix_name***.

➢ Zweck

Löscht ein zuvor definiertes Synonym.

➢ Syntax

DROP SYNONYM *synonymname*

➢ Anwendung

Wird eine Tabellegelöscht, so bleibt das Synonym bis zur ausdrücklichen Anwendung der DROP SYNONYM Anweisung bestehen.

➢ Beispiel

```
DROP SYNONYM syscolumns ;
```

In diesem Beispiel werden die Tabellen ***syscolumns*** und ***SysColumns*** gelöscht, da ***syscolumns*** ein Synonym für ***SysColumns*** ist (beide Tabellen werden durch **sqltools** automatisch mit jedem Erstellen einer Datenbank erstellt).

➢ Zweck

Löscht eine Tabelle zusammen mit den dazugehörigen Indizes und den Daten.

➢ Syntax

DROP TABLE {*tablename* xor *synonymname*}

➢ Anwendung

Wird ein Synonym von der Anweisung **DROP TABLE** gelöscht, wird die Tabelle ebenfalls gelöscht.

Gilt die Anweisung **DROP TABLE** für eine Tabelle, werden die Synonyme der Tabelle nur gelöscht, wenn die Anweisung **DROP SYNONYM** verwendet wird.

➢ Beispiel

```
DROP TABLE TABLE1;
```

Löscht die Tabelle TABLE1 und deren Indizes und Daten.

➢ Zweck

Löscht eine mit der Anweisung DEFINE erstellte Tabelle, jedoch nicht die dazugehörigen Daten- und Indexdateien.

➢ Syntax

UNDEFINE TABLE {*tablename* xor *synonymname*}

➢ Beispiel

```
UNDEFINE TABLE TABLE1;
```

Löscht die Tabelle TABLE1, die mit der Anweisung DEFINE TABLE TABLE1 erstellt wurde.

➢ Zweck

Datenbankabfrage.

➢ Syntax

SELECT Select clause From clause [Where clause] [Group by clause] [Having clause] [Order by clause]

➢ Anwendung

Sie können die Tabellen in der aktuellen oder einer anderen Datenbank abfragen.

➢ Beispiel

```
SELECT customer_name
FROM customers
WHERE turn_over > 250
ORDER BY country ;
```

Diese Abfrage liest aus der Tabelle ***customers*** Kunden mit einem jährlichen Umsatz von über 250 und gibt ihre Namen sortiert nach Land aus.

Benutzt in den **SELECT** und **INSERT** Anweisungen.

➢ Syntax

[{ALL xor DISTINCT xor UNIQUE}] {Expression [[AS] *Display label*] [Expression [[AS] *Display label*]] ^{n}

Mit der Bedingung SELECT bestimmen Sie die zu wählenden Daten und ob doppelt vorkommende Werte auszulassen sind.

➢ Anwendung

ALL	Alle gewählten Werte werden zurückgegeben, auch wenn Sie doppelt aufscheinen.
DISTINCT	Entfernt doppelt erscheinende Zeilen vom Abfrageergebnis.
UNIQUE	Synonym für DISTINCT.

➢ Beispiel

```
SELECT customer_name
FROM customers
WHERE turn_over > 250
ORDERBY country ;
```

Diese Abfrage sucht in der Tabelle *customers* nach allen Kunden mit einem jährlichen Umsatz von über 250 und gibt die Namen sortiert nach Land aus.

```
SELECT order_date, COUNT(*), paid_date - order_date
FROM orders
GROUP BY 1, 3
```

Diese Abfrage gibt die Auftragsnummer, Anzahl der Aufträge und die Differenz zwischen Zahlungs- und Auftragsdatum, gruppiert nach Auftragsdatum und Zeitunterschied (zwischen Zahlungs- und Auftragsdatum) aus.

Benutzt in der **SELECT** Anweisung.

➢ Syntax

{ [{*tablename* xor *synonymname* xor *tablealias*}.] *columnname*
xor NULL
xor Literal number
xor Quoted string
xor User
xor Aggregate expression}

➢ Beispiel

```
'Cordwainer'
```

Die Zeichenkette ***'Cordwainer'*** ist ein Unterausdruck.

Benutzt in den **SELECT** und **DELETE** Anweisungen.

➢ Zweck

Listet die Tabelle oder Tabellen mit gewählten Daten.

➢ Syntax

FROM {Table name xor Synonym name} [[AS] *Table alias*]
[,{Table name xor Synonym name} [[AS] *Table alias*]] [n]

➢ Beispiel

```
SELECT customer_name, order_num
FROM customers c, orders o
WHERE c.customer_num = o.customer_num ;
```

Diese Anweisung holt Daten von den Tabellen ***customers*** und ***orders***, die Tabellen bekommen Aliasse.

Benutzt in den **SELECT**, **DELETE** und **UPDATE** Anweisungen.

➢ Zweck

Bestimmt die Suchbedingung(en).

➢ Syntax

WHERE Condition [AND Condition] ^{n}

➢ Beispiel

```
SELECT customer_name
FROM customers
WHERE last_order_date < '28/07/1993'
ORDERBY country ;
```

In diesem Beispiel ist die Suchbedingung das letzte Auftragsdatum.

Benutzt in dem **SELECT** Anweisung.

➢ Zweck

Ergibt für jede Gruppe eine einzelne Zeile von Ergebnissen.

➢ Syntax

GROUP BY {Table name xor Synonym name} . *Column name* xor *Select number*}
[,{Table name xor Synonym name} . *Column name* xor *Select number*}]^{n}

➢ Anwendung

Eine Gruppe ist ein Satz von Reihen, die für jede angeführte Spalte die gleichen Werte haben.

Die Variable "select number" ist eine Ganzzahl, die die Position einer Spalte in der Bedingung SELECT angibt.

➢ Beispiel

```
SELECT order_date, COUNT(*), paid_date - order_date
FROM orders
GROUPBY order_date, 3 ;
```

Die Ergebnisse sind gruppiert nach ***order_date*** und ***paid_date - order_date***.

Benutzt in dem **SELECT** Anweisung.

➢ Zweck

Unterlegt Gruppen einer Bedingung oder mehreren Bedingungen

➢ Syntax

HAVING Condition

➢ Beispiel

```
SELECT customer_num, call_dtime, call_code
FROM cust_calls
GROUP BY call_code, 2 , 1
HAVING customer_num < 42 ;
```

Diese Abfrage ergibt Tabellen mit call_code, call_dtime und customer_num und gruppiert sie nach call_code, für alle Anrufe von Kunden, deren Kundennummer unter 42 ist.

Benutzt in dem **SELECT** Anweisung.

➢ Zweck

Sortiert Abfrageergebnisse nach Werten ein einer oder mehreren Spalten.

➢ Syntax

ORDER BY {*Table name* . xor *Synonym name* .} *Column name* xor *Select number* xor *Display label*} [,{*Table name* . xor *Synonym name* .} *Column name* xor *Select number* xor *Display label*}]^{n}

➢ Anwendung

Die Variable "select number" ist eine Ganzzahl, die die Position einer Spalte in der Bedingung SELECT angibt.

➢ Beispiel

```
SELECT customer_name
FROM customers
WHERE turn_over > 250
ORDERBY country ;
```

Das Abfrageergebnis ist nach Land sortiert.

➢ Zweck

Löscht eine Zeile oder mehrere Zeilen aus einer Tabelle.

➢ Syntax

DELETE FROM {*Table name* xor *Synonym name* }
[WHERE {Condition xor CURRENT OF *Cursor name*}]

➢ Beispiel

```
DELETE FROM customers WHERE last_order_date<1992;
```

Löscht aus der Tabelle *customers* Reihen, deren letztes Auftragsdatum vor 1992 ist.

➢ Zweck

Fügt einer Tabelle eine oder mehrere Reihen hinzu.

➢ Syntax

INSERT INTO {*Table name* xor *Synonym name* } [(Column name [,Column name] ^{n})] {Values clause xor Select clause}

➢ Beispiel

```
INSERT INTO Pets VALUES ('Socks', 'Cat', 'M') ;
```

Die Anweisung fügt in die Tabelle ***Pets*** die Werte ***'Socks'***, ***'Cat'*** und ***'M'*** ein.

Benutzt in dem **INSERT** Anweisung.

➢ Syntax

VALUES ({*Variable name* : *Indicator variable* xor NULL xor Literal number xor Quoted string xor Literal Timestamp xor Literal date xor Literal time} [,{*Variable name* : *Indicator variable* xor NULL xor Literal number xor Quoted string xor Literal Timestamp xor Literal date xor Literal time}]^{n})

➢ Anwendung

Mit der Bedingung VALUES können Sie jeweils nur eine einzelne Zeile einfügen. Jeder Wert nach dem Schlüsselwort VALUES wird der entsprechenden Spalte zugewiesen, die in der Bedingung INSERT INTO angeführt ist (oder Spalten nacheinander, falls keine Liste von Spalten definiert ist).

➢ Beispiel

```
INSERT INTO Pets VALUES ('Socks', , 'M') ;
```

Diese Anweisung fügt den Wert *'**Socks**'* in die erste Spalte und den Wert *'**M**'* in die dritte Spalte der Tabelle ***Pets*** ein. Die zweite Spalte bleibt unberührt.

➢ Zweck

Ändert den Wert in einer oder mehreren Spalten oder einer oder mehrere Reihe(en) einer Tabelle.

➢ Syntax

UPDATE {*Table name* xor *Synonym name*} SET Set clause [WHERE {Condition xor CURRENT OF *Cursor name*}]

➢ Beispiel

```
UPDATE Catalog SET item_price = 10 WHERE item_type =
'L' ;
```

Diese Anweisung ändert die Preise aller Artikel des Typs ***'L'*** in der Tabelle ***Katalog*** auf 10.

Benutzt in dem **UPDATE** Anweisung.

➢ Syntax

{Column name = {Constant xor Select statement xor NULL} [,Column name = {Constant xor Select statement xor NULL}]^{n}
xor {(*Column name* [,*Column name*]^{n} xor *} = (Select statement)

➢ Beispiel

```
UPDATE Catalog SET item_price = 10
```

In dieser Anweisung ändert die **SET**-Bedingung den Preis *item_price* auf 10.

Benutzt in dem **SELECT** Anweisung oder in einem Ausdruck.

➢ Syntax

{COUNT(*)
xor
{MIN xor MAX xor SUM xor AVG xor COUNT} ([{DISTINCT xor UNIQUE}] {*Table name* xor *Synonym name* xor *Table alias*} . *Column name*)
}

➢ Beispiel

```
SELECT COUNT (DISTINCT item_type) FROM Catalog ;
```

Diese Anweisung gibt die Anzahl der verschiedenen Artikeltypen in der Tabelle ***Catalog*** aus.

Datentypen

Dieser Abschnitt beschreibt die von der SQL-Sprache unterstützten Datentypen und die entsprechenden Definitionen in C. Die Beschreibungen entsprechen den von CREATE TABLE verwendeten SQL-Typen. Sie sind systemeigene Entsprechungen. Für den Import von Dateien, die außerhalb der C-ISAM-Datenbank mit dem Befehl DEFINE TABLE erstellt wurden, werden die Unterschiede aufgezeigt.

In der nachfolgenden Tabelle gelten die kursiv gestellten Typen in der Spalte C-Sprachentyp nur für die Anwendung mit CREATE TABLE.

SQL-Typ in der Datenbank	SQL-Typ in ODBC	C-Sprachentyp
bit	SQL_BIT	*unsigned char notnull_data;* unsigned char data;
byte	SQL_TINYINT	*unsigned char notnull_data;* unsigned char data;
char(maxlength) 1 <= maxlength <= 32511	SQL_CHAR	char data[maxlength];
varchar(maxlength) 1 <= maxlength <= 32511	SQL_VARCHAR	char data[maxlength];
binary(maxlength) 1 <= maxlength <= 32511	SQL_BINARY	*unsigned char notnull_data;* unsigned char data[maxlength];
varbinary(maxlength) 1 <= maxlength <= 32511	SQL_VARBINARY	*unsigned char size_data[2];* *unsigned char data[maxlength];*
smallint	SQL_SMALLINT	short data; /* 2 bytes */
longint	SQL_INTEGER	long data; /* 4 bytes */
real	SQL_FLOAT	float data; /* 4 bytes */
double	SQL_DOUBLE	double data; /* 8 bytes */
decimal(prectot, precdec) 1<=prectot<=32 et 0<=precdec<=prectot	SQL_DECIMAL	char data[(prectot+1)/2+1];
date	SQL_DATE	unsigned long data;
time	SQL_TIME	unsigned long data;
timestamp	SQL_TIMESTAMP	unsigned long data;

➢ Der Typ Bit

Dieser Typ entspricht dem Typ SQL_BIT in ODBC. Er speichert die Binärwerte 0 und 1 oder den Wert Null.

Wird dieser Datentyp mit CREATE TABLE verwendet, so wird in der erstellten Datei ein 2-Byte Block reserviert, um zwischen dem Nullwert und 0 oder 1 zu unterscheiden. Wird so eine Datei in der Definition eines Schlüssels verwendet, werden die zwei Byte im Schlüssel verwendet.

Wird der Bit-Datentyp mit DEFINE TABLE verwendet, so wird nur ein Byte reserviert. Es wird nun nicht mehr zwischen dem Wert 0 und dem Nullwert unterschieden. In DEFINE TABLE kann dieser Feldtyp daher nicht null sein.

Die folgende Tabelle zeigt die gespeicherten Werte. Die kursiv dargestellten Daten beziehen sich nur auf CREATE TABLE.

Feld Bit-Typ	***notnull_data***	**Datenwert**
0	*1*	0
1	*1*	1
Null	*0*	*0*

➢ Der Typ Byte

Dieser Typ entspricht dem Typ SQL_TINYINT in ODBC. Er speichert Werte von 0 bis 255 oder Null.

Wird dieser Datentyp mit CREATE TABLE verwendet, so wird in der erstellten Datei ein 2-Byte Block reserviert (wie beim Typ **bit**), um den Nullwert von den anderen Werten zu unterscheiden. Wird so eine Datei in der Definition eines Schlüssels verwendet, werden die zwei Byte im Schlüssel verwendet.

Wird der Byte-Datentyp mit DEFINE TABLE verwendet, so wird nur ein Byte reserviert. Es wird nun nicht mehr zwischen dem Wert 0 und dem Nullwert unterschieden. In DEFINE TABLE kann dieser Feldtyp daher nicht null sein.

Die folgende Tabelle zeigt die gespeicherten Werte. Die kursiv dargestellten Daten beziehen sich nur auf CREATE TABLE.

Byte-Typ Feld	*notnull_data*	Datenwert
0	*1*	0
1	*1*	1
...	*1*	...
255	*1*	255
null	*0*	*0*

➢ Der Typ char

Dieser Typ entspricht dem Typ SQL_CHAR in ODBC. Er speichert von 1 bis 32511 Zeichen.

Werden Daten dieses Typs in ein Feld eingefügt, so werden all nicht-signifikanten Leerzeichen am Ende der Kette gelöscht und die entsprechenden Byte auf '\0' (ASCII-Code 0) gesetzt. Beim Lesen der Daten aus diesen Feldern wird die Zeichenkette automatisch mit Leerzeichen (ASCII-Code 32) zu ihrer maximalen Größe vervollständigt. Wird eine leere Zeichenkette in einem Feld des Typs **char** gespeichert, wo wird ein einzelnes Leerzeichen ins Feld geschrieben, um es von Null zu unterscheiden.

Feldtyp char(10)	Datenwert
‘’	0x32000000000000000000
‘ ‘	0x32000000000000000000
‘A ‘	0x41000000000000000000
‘AaBb’	0x41614262000000000000
null	0x00000000000000000000

➢ Der Typ varchar

Dieser Typ entspricht dem Typ SQL_VARCHAR in ODBC. Er wird praktisch in der gleichen Weise wie der Typ **char** verwendet. Er speichert auch von 1 bis 32511 Zeichen.

Im Unterschied zum Typ **char** werden beim Einfügen der Daten die nicht-signifikanten Leerzeichen am Ende der Kette nicht gelöscht; die entsprechenden Byte in der Datei werden mit dem Zeichen '\0' gefüllt. Beim Lesen von Daten von diesen Feldern holt ODBC die nicht geänderte Zeichenkette. Wie für den Typ **char**, wird eine leere Kette als einzelnes Leerzeichen gespeichert, um sie von Null zu unterscheiden.

Feldtyp varchar(10)	Datenwert
‘’	0x32000000000000000000
‘ ‘	0x32000000000000000000
‘A ‘	0x41202020200000000000
‘AaBb’	0x41614262000000000000
null	0x00000000000000000000

➢ Der Typ bynary

Dieser Typ entspricht dem Typ SQL_BINARY in ODBC. Er speichert von 1 bis 32511 Zeichen.

Wird dieser Datentyp mit CREATE TABLE benutzt, so wird ein extra Byte (notnull_data) verwendet. Dieses Byte kann auf 0 oder 1 gesetzt werden, um den Wert Null von einer leeren Zeichenkette zu unterscheiden. Wird ein Feldtyp **binary** in der Definition eines Schlüssels verwendet, so wird das Byte notnull_data mit den Datenbyte im Schlüssel verwendet..

Bei Verwendung mit DEFINE TABLE wird kein extra Byte hinzugefügt. Nur die signifikanten Byte werden gespeichert. Der Wert 0 kann nicht vom Nullwert unterschieden werden, daher kann dieser Feldtyp nicht Null sein.

Werden Daten in diesen Feldtyp eingefügt, so werden die entsprechenden Byte in der Datei mit dem Zeichen '\0' vervollständigt. Beim Lesen dieses Datentyps greift ODBC auf alle Byte zu.

Die folgende Tabelle zeigt die gespeicherten Werte. Die kursiv dargestellten Daten beziehen sich nur auf CREATE TABLE.

Feldtyp binary(10)	***notnull_data***	**Datenwert**
0x	*1*	0x00000000000000000000
0x00	*1*	0x00000000000000000000
0x1234	*1*	0x12340000000000000000
null	*0*	*0x00000000000000000000*

➢ Der Typ varbinary

Dieser Typ entspricht dem Typ SQL_VARBINARY in ODBC. Er speichert von 1 bis 32511 Zeichen.

Dieser Typ kann nur mit dem Befehl CREATE TABLE, nicht mit DEFINE TABLE verwendet werden. In diesem Datentyp werden zwei extra Byte verwendet. Dieses Byte können die Länge der Binärdaten plus Eins als kurze Ganzzahl speichern. Der Wert 0 entspricht einer Binärnull und der Wert 1 einer binary der Länge 0.

Wie beim Typ **binary** wird bei der Dateneingabe in diesen Feldtyp der entsprechende Block in der Datei mit dem Zeichen '\0' vervollständigt. Beim Lesen dieses Feldtyps greift ODBC nur auf die Byte mit der binären Länge zu. Wird der Feldtyp **varbinary** in der Definition eines Schlüssels verwendet, so sind die size_data Byte nicht mit den Datenbyte im Schlüssel enthalten.

Feldtyp varbinary(10)	size_data	Datenwert
0x	0x0001	0x00000000000000000000
0x00	0x0002	0x00000000000000000000
0x1234	0x0003	0x12340000000000000000
null	0x0000	0x00000000000000000000

➢ Der Typ smallint

Dieser Typ entspricht dem Typ SQL_SMALLINT in ODBC. Er speichert in 2 Byte Ganzzahlen im Bereich -32767 bis 32767. Der Wert –32768 ist für ein Nullfeld reserviert.

Feldtyp smallint	Datenwert
-32767	**-32767**
0	**0**
30000	**30000**
null	**-32768**

➢ Der Typ logint

Dieser Typ entspricht dem Typ SQL_INTEGER in ODBC. Er speichert Ganzzahlen im Bereich -134217727 bis 134217727 in 4 Byte. Der Wert –134217728 ist für ein Nullfeld reserviert.

Feldtyp longint	Datenwert
-32767	-32767
0	0
134217	134217
null	-134217728

➢ Der Typ real

Dieser Typ entspricht den Typen SQL_REAL und SQL_FLOAT in ODBC. Er speichert floating Point Nummern in Maschinenformat in 4 Byte.

Feldtyp real	Datenwert
-25	(float)-25.0
0	(float)0.0
3.1415	(float)3.1415
null	non-significant value

➢ Der Typ double

Dieser Typ entspricht dem Typ SQL_DOUBLE in ODBC. Er speichert floating Point Nummern in Maschinenformat in 8 Byte.

Feldtyp double	Datenwert
-25	(double)-25.0
0	(double)0.0
3.1415	(double)3.1415
null	non-significant value

➢ Der Typ decimal

Dieser Typ entsrpicht dem Typ SQL_DECIMAL in ODBC. Er speichert fixed Point Nummern.

Auf diesen Typ müssen zwei in Klammern gesetzte Parameter folgen, "c1 decimal(n, m)", wobei n die Anzahl der Stellen insgesamt und m die Anzahl von decimal ist. Ist m nicht definiert, wird der Standardwert 0 verwendet.

Für Lese/Schreiboperationen werden in diesem Feldtyp die C-ISAM-Funktionen stdecimal() und lddecimal() verwendet.

➢ Der Typ date

Dieser Typ entspricht dem Typ SQL_DATE in ODBC. Er speichert ein Datum als die Anzahl der Tage seit dem ersten Tag des Jahres 0 als eine 4-Byte-Ganzzahl. Um ein Datum einzufügen oder in SQL-Befehlen zu testen, verwenden Sie die ODBC-Schreibweise ({d ‘AAAA-MM-JJ’ }).

Feldtyp date	Datenwert
{ d ‘0000-01-01’ }	0
{ d ‘1997-02-17’ }	729438
null	-134217728

➢ Der Typ time

Dieser Typ entspricht dem Typ SQL_TIME in ODBC. Er speichert die Daten als Anzahl der Sekunden im Tag in einer 4-Byte-Ganzzahl. Um eine Zeit einzufügen oder in SQL-Befehlen zu testen, verwenden Sie die ODBC-Schreibweise ({ t ‘hh :mm :ss’ }).

Feldtyp time	Datenwert
{ t ’00 :00 :00’ }	0
{ t ’13 :40 :10’ }	49210
null	-134217728

➢ Der Typ timestamp

Dieser Typ entspricht dem Typ SQL_TIMESTAMP in ODBC. Er speichert die Zeit als die Anzahl der Sekunden vom 1. Januar 1970 (ähnliche der Funktion time() in C). Der Maximalwert entspricht dem Datum 5. Februar 2036, 00.00 h. Um Timestamp einzufügen oder in SQL-Befehlen zu testen, verwenden Sie die ODBC-Schreibweise ({ ts ‘AAAA-MM-JJ hh :mm :ss’ }).

Feldtyp timestamp	Datenwert
{ ts ’1970-01-01 00 :00 :00’ }	0
{ ts ’1997-02-17 13 :40 :10’ }	856186810
null	-134217728

INDEX

U

V

W

Z