USER MANUAL

Tun SQL™

Client/Server for Windows

Version 2.00

In continuing our efforts to improve our products, keep up with current technology, and incorporate user's suggestions, ESKER may change the look and the characteristics of the product described in this manual at any time.

ESKER makes no warranty of any kind, expressed or implied, with regard to the contents of this manual. This is a purely technical document and does not represent a contractual obligation of any kind. ESKER shall not be held liable for incidental or consequential damages in connection with the use of the programs described herein.

PREFACE

Tun SQL is a complete set of WINDOWS drivers and UNIX servers which enable a PC to work conveniently in Client / Server mode on UNIX databases (Informix, Oracle, Sybase, Progress, DB2). **Tun SQL** uses the ODBC architecture as defined by Microsoft.

Tun SQL is a complementary product to the **Tun KERNEL**, **Tun NET** and **Tun EMUL** software range. It fits into the Tun range as follows:

	WINDOWS	MS-DOS
Tun KERNEL	TCP/IP protocol stacks for Windows 3.x only	TCP/IP protocol stacks for MS-DOS (TSR)
Tun NET	TCP/IP applications (NIS, NFS Client and Server, PING, Printer redirection and sharing, FTP Client and Server, TELNET, RSH Client and Server, TAR, WALL, TFTP, TIME), and electronic mail and fax applications	TCP/IP applications for MS-DOS (NFS, Printer sharing, FTP, TELNET, TAR ...)
Tun EMUL	Comprehensive terminal emulator (asynchronous emulation, IBM 3270, IBM 5250)	Comprehensive terminal emulator for MS-DOS (asynchronous emulation)
Tun SQL	ODBC drivers for Client-Server mode under TCP/IP (Oracle, Informix, Sybase DBMS, Progress, DB2) and database revamping tools	N/A
TCP/IP Network Services	Browser NIS, Printer redirection and sharing (LPR, LPD)	N/A

Tun SQL is delivered as standard with the product **Tun PLUS** which combines all the above-mentioned programs in one software package.

Tun SQL may be installed separately from **Tun PLUS** with the complementary products **Tun KERNEL** (for Windows 3.x) and **TCP/IP Network Services**. In any case, the installation procedure of **Tun PLUS** automatically proposes the optional installation of**Tun SQL.**

TABLE OF CONTENTS

PART 1
PRESENTATION AND INSTALLATION

CHAPTER 1 - INTRODUCTION TO Tun SQL

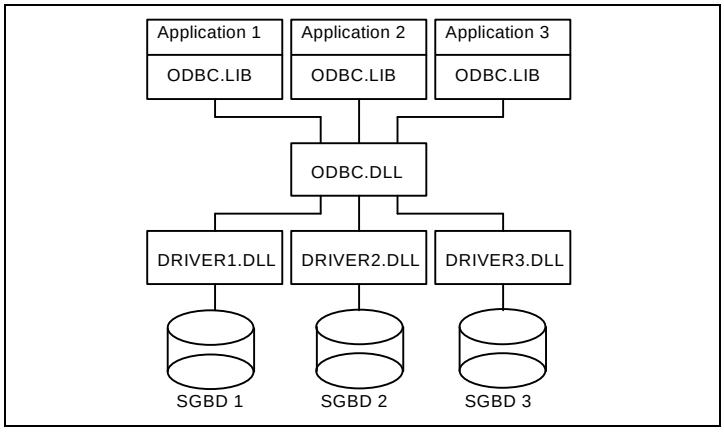
THE ODBC MECHANISM

In the world of databases, programmers traditionally use a mechanism called "Embedded SQL" to provide an interface between their applications and a specific database. The Embedded SQL mechanism makes it possible to insert SQL requests into programs written in COBOL or C. It offers the advantage of making applications more widely portable on different machines.

The Embedded SQL mechanism, however, presents several disadvantages:

- There are as many Embedded SQLs as there are DBMS engines on the market. The applications using SQL can only interface with one DBMS at a time. They have to be rewritten or at least modified if they are to be interfaced with other databases. For applications which are to interface with all the databases on the market, it is inconceivable to use the Embedded SQL mechanism.
- The Embedded SQL mechanism is relatively undeveloped, quite restrictive and difficult to use. It does not allow databases to be used to full advantage, and it is sometimes preferable to use the API provided by the DBMS directly.

To compensate for the disadvantages of the Embedded SQL mechanism, Microsoft conceived a new approach based on the ODBC mechanism (Open Database Connectivity) which is based on WOSA (Windows Open System Architecture).



ODBC is a well-defined set of C functions which makes it possible to retrieve or update data in a DBMS. These functions have been assembled in a DLL (Dynamic Link Library) which can be used by any Windows application. The functions of the ODBC DLL (ODBC.DLL) analyze SQL requests and has them processed by ODBC drivers whose job it is to convert the calls to suit the particular API of the DBMS you wish to use. An ODBC driver allows you to view the DBMS interface and to enable an application to use it just like any other ODBC-compliant DBMS.

Microsoft supplies the ODBC.DLL library and all the tools required to use it; however, they do not supply ODBC drivers for all the DBMSs on the market. Microsoft is satisfied with supplying drivers for its proprietary storage systems (Excel, Word, Access...).

Specific ODBC drivers for databases may be supplied directly by the DBMS publisher or by third parties who specialize in this domain (ESKER).

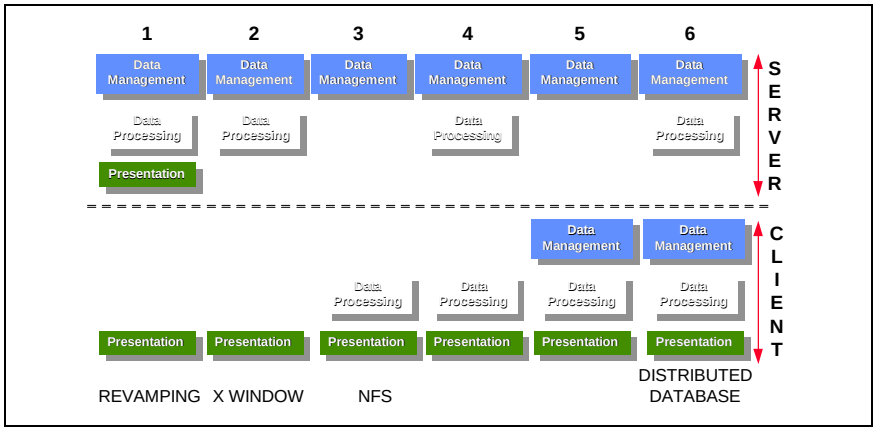
Finally, the ODBC mechanism provides maximum interoperability. A simple Windows application may access different management systems without necessarily having been designed with this in mind. ODBC allows developers to program, compile and deliver their program without having to worry about the DBMS on which the program will be used. The users have only to provide the right driver so the application on which they are working will cooperate with the DBMS of their choice.

ODBC is a valuable mechanism for multi-domain applications such as spreadsheets, word processing applications, and development tools which can manipulate information from any DBMS without knowing *a priori* which DBMS is being used.

THE CLIENT/SERVER MODEL

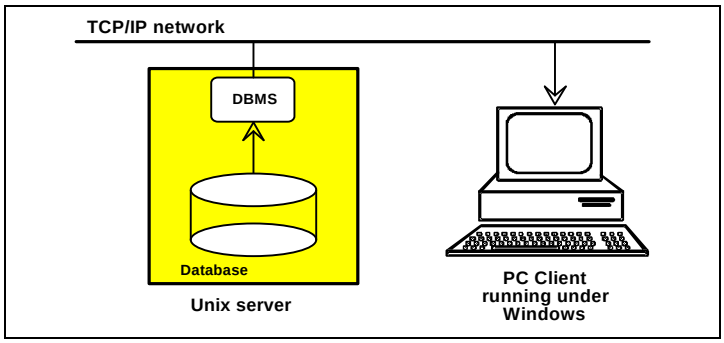
For some years, Client/Server has been the term on everyone's lips in the computing industry. In the widest sense, the Client/Server model is a conception of computing in which at least two units cooperate to supply a particular service.

The "Gartner Group" has identified six principal types of Client/Server mode applications which they have classified according to the number of functions performed by the client or the server:



The applications which make the least demands on the "client" are "revamping" applications or X-WINDOWS servers. The applications which make the most demands on the "client" are those which use distributed databases.

When the computing industry speaks about the Client/Server model, it is very often referring to applications concerned with distributed databases. The most common illustration of this mode is shown in the following schematic:



A "client" PC has a conventional management system which works in graphic mode. The data is stored centrally on an UNIX server and managed by a Relational Database Management System (RDBMS). To retrieve or update data, the client sends a SQL request to the server which executes it and returns the reply to the client over the network.

The principal advantages of this architecture are the following:

- The end-user has a graphic, user-friendly Man/Machine interface on his PC running under the Windows operating system.
- The PC can be used for tasks other than the normal applications (office applications, calculations, personal applications...).
- While having his own machine, the user can have access to centralized data on a server at the same time as other users.
- The server is relieved of the applications part of computing and all its power can be concentrated on serving data. There is a balanced distribution of the workload between the client and the server.
- The network is only used to send essential application data; it is not overloaded with presentation information.

The SQL Client/Server mode as described above is a modernized version of the transactional mode as may still be encountered in a Mainframe-synchronous terminal environment.

ODBC AND THE SQL CLIENT/SERVER MODEL

Insofar as the ODBC mechanism facilitates access to all systems of data management, it can also be used on a remote database. In this context, ODBC enables a Windows application to use any DBMS of whatever origin. It also enables multi-domain applications such as "Excel", "Word" or "Access" to use the company's centralized information. In a Client/Server context, the ODBC mechanism allows you to increase the number of applications which can use the company's data.

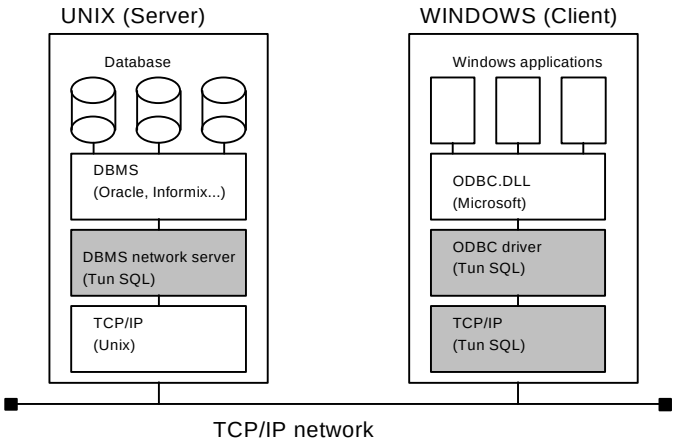
In any case, the implementation of such architecture today is no simple matter. Indeed, a user who already has a local PC network, a UNIX server and a DBMS must also acquire the following components to set in operation an SQL Client/Server architecture using ODBC:

- The network server part of the DBMS (Informix Net, Sql Net...)
- The PC client part of the DBMS (Informix Net PC, Sql Net PC...)
- The appropriate ODBC driver
- **Winsock**-compatible TCP/IP stacks for the PC

DBMS publishers always supply the first two components. This is not always the case for the ODBC driver and is never the case for the TCP/IP stacks. It is therefore necessary to procure the last two components elsewhere paying particular attention to the question of compatibility.

Tun SQL

Tun SQL was conceived to solve the problem definitively. It integrates all the components mentioned above, as well as powerful database revamping functionality and a virtual ODBC driver for access to the revamped databases, into one homogenous software package. Even the network components of the DBMS (client and server) are included in **Tun SQL**. This represents a considerable saving when it comes to equipping a lot of PCs. It especially makes things very simple when it comes to implementing a SQL Client/Server architecture.



The principal features of **Tun SQL** are as follows:

One ODBC driver only for most of the Unix DBMSs on the market

To limit the number of software products required for the PC client, **Tun SQL** features one sole ODBC driver to access the following Unix DBMSs indifferently:

- Oracle version 7.
- Informix version 5 and version 7.
- Sybase version 10.
- DB2 version 2.
- Progress version 6.

Database revamping

Tun SQL, with the help of an integrated user-friendly application, makes it possible to redefine the tables in a database and make them more accessible to the end-user (through customized table reorganization, modification of table and field names, and preset functions).

A virtual ODBC driver for revamped databases

To use a database that has been redefined by revamping, **Tun SQL** includes a virtual ODBC driver which translates requests made to virtual tables into requests that a normal ODBC driver can handle.

The server part of the DBMS is included in Tun SQL

The server part of each of the DBMS is installed under Unix and is supplied as standard with **Tun SQL** for the following operating systems:

- 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

This feature saves **Tun SQL** users the cost of the server part of the DBMS they use.

TCP/IP stacks delivered as standard

Like all the software in the Tun range, **Tun SQL** is delivered with ESKER's TCP/IP stack as standard. The stack has an excellent performance record and has been tested with all of **Tun SQL**'s components. The inclusion of stack in the **Tun SQL** package saves the user the trouble of procuring them elsewhere.

Simplicity of installation and administration

The purpose of **Tun SQL** is to facilitate the implementation of a Client/Server architecture based on Windows and Unix. Consequently, **Tun SQL** has a simple installation procedure for Unix and Windows and comes with complete documentation.

In addition to the ODBC driver, two Windows applications are supplied for testing and implementing the Client/Server:

- **Tun DB Show** is used for testing the Client/Server connection from end to end. The application can find out from the Unix server which DBMSs are installed and query some DBMSs to find out which databases they contain.
- **Tun DB Script** can execute SQL batch files under Windows to create databases on the remote DBMS.

In addition to the security offered by Unix and the different DBMSs, **Tun SQL** supplies a mechanism which can deny some Windows applications access to particular sensitive databases.

Lastly, the integration of the NIS (Network Information Service) into **Tun SQL** allows the centralized management of network resources and facilitates access to remote resources. For more information on the NIS, please consult the **TCP/IP Network Services** user manual.

Detailed examples

In order to familiarize the user with the ODBC mechanism in the Client/Server environment, **Tun SQL** is delivered with a sample database which can be directly uploaded from a Windows station to a remote DBMS. **Tun SQL** comes with several examples of applications which use this database:

- A query sample using Microsoft Excel
- A query sample using Microsoft Winword
- A Visual Basic application for viewing the structure of a database

Tun SQL driver conformity

The **Tun SQL** driver supports all the level 1 functions. It supports some of the level 2 functions. The "Microsoft ODBC Cursor Library" is supplied with the driver. Although this library only supports static and "forward only" cursors, this is sufficient for many applications.

CHAPTER 2 - INSTALLATION UNDER WINDOWS

PACKAGE CONTENTS

Please make sure your **Tun SQL** package contains the following:

- The **Tun SQL** user manual (Client/Server for Windows).
- The **TCP/IP Network Services** manual (NIS, printer sharing and redirection).
- The **Tun KERNEL** manual (TCP/IP Communication for Windows 3.x).
- A CD-ROM.
- A reply coupon to obtain the UNIX part of **Tun SQL** for the platform and DBMS of your choice.
- A user license.
- A sealed envelope containing a serial number and an activation key. Miscellaneous technical bulletins (if applicable).
- A registration card.

> **Note**: Opening the sealed envelope indicates that you have accepted the terms and conditions of the User license (indicated on the sealed envelope) for using **Tun SQL**.

HARDWARE AND SOFTWARE REQUIREMENTS

To use **Tun SQL** for all Windows effectively you will need the following equipment:

- A 100% PC 486 compatible micro-computer or Pentium.
- 8-16 MB of RAM.
- Windows 3.x or 95.
- TCP/IP stack compatible with the Winsock interface (e.g. **Tun KERNEL** or other Winsock-compatible stacks).
- A UNIX server with an RDBMS (Oracle, Informix, Sybase, DB2 or Progress).

PRODUCT INSTALLATION

The following instructions may be ignored if you have acquired Tun SQL as part of the Tun PLUS package since the installation procedure of the latter product will automatically propose the installation of Tun SQL.

The following procedure describes the installationunder Windows of the **Tun SQL** package:

1. Insert the CD-ROM into the CD-ROM drive (generally, drive D).
Depending on your Windows version, follow one of these steps:

Under Windows 3.x or Windows NT3.51, choose one of the following methods:

- From the **Program Manager**, select **File➔Run** and type the following command:
  D:\INSTALL.EXE

- From the **File Manager**, open the drive containing the CD-ROM (drive D) et double-click on the file INSTALL.EXE.

Under Windows 95 or Windows NT4, choose one of the following methods:

- Click on the **Add/Remove Programs** icon in the **Control Panel** window. Follow the installation prompts that appear, using the file INSTALL.EXE on the CD-ROM.

- From a DOS prompt, type the following command:

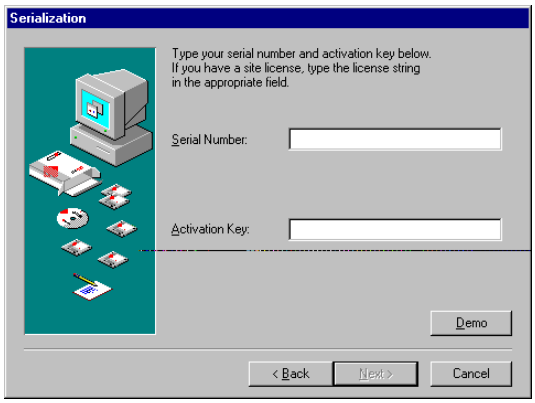
  D:\INSTALL.EXE

- From the **Start** menu, select **Run** and type the above command.

- From the **Windows Explorer**, open the drive containing the CD-ROM (drive D) and double-click on the file INSTALL.EXE.

Note:	If you are using the **Tun SQL** installation disks instead of the CD-ROM, insert the disk labeled "Tun Setup Disk 1/2" into the disk drive (generally drive A), and follow the same procedure with the following command: **A:\SETUP.EXE**

2. After the Welcome dialog box, you will see the following installation window.

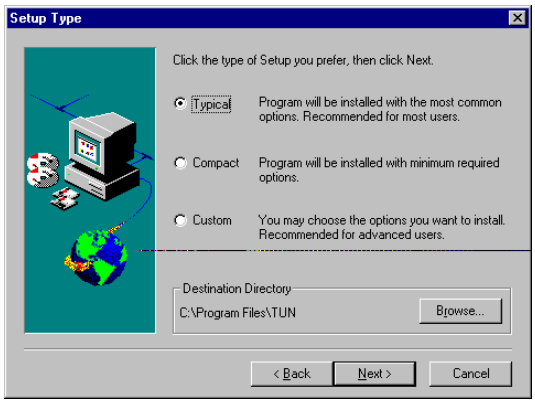


Enter the serial number and the activation key of the software. This information can be found in the sealed envelope accompanying the software.

Click the button **Demo** if you wish to install the demonstration version of **Tun PLUS**. A serial number and activation key will be proposed for the demo installation.

Then click the button **Next**.

3. If the serial number and activation key are correct, the following window will appear:



This dialog box lets you choose the type of installation you want and the installation directory.

Type of installation

There are three types of installation:

- **Typical**: installs the necessary components for normal usage of **Tun SQL**.
- **Compact**: installs the minimum number of components for **Tun SQL** to function.
- **Custom**: allows you to choose the components you wish to install.

Installation directory

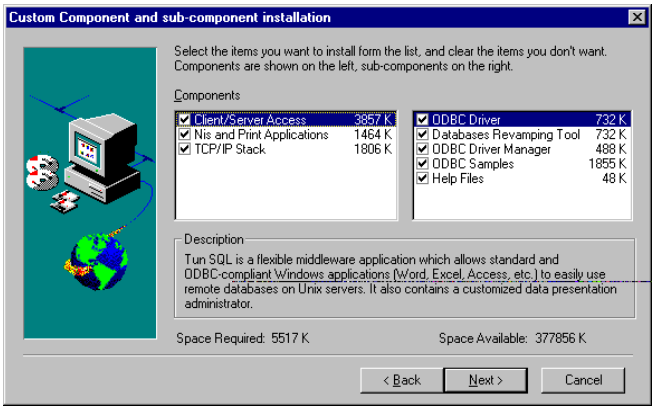
In Windows 3.x, the default directory for the installation of **Tun SQL** is C:\TUN.

In Windows 95, the default directory is C:\Program Files\TUN.

The directory may be changed by clicking the button **Browse...**.

Click the button **Next** when you are ready.

4. If you chose the custom installation, the following dialog box will appear:



In Windows 95, the option **TCP/IP Stack** is not available, since **Tun SQL** uses Microsoft's TCP/IP stack.

Select the check boxes corresponding to the components or subcomponents you wish to install.

Client/Server Access

This component includes three subcomponents:

- ODBC Driver: **Tun SQL**'s ODBC driver allows access to UNIX DBMSs such as Oracle version 7, Informix versions 5 and 7, Sybase version 10, DB2/6000 version 3.1 and Progress version 6.
- Databases Revamping Tool: the database revamping tool creates virtual tables more adapted to the needs of the end-user; the **Tun SQL** ODBC driver lets the user access the redefined database.
- ODBC Driver Manager: basic ODBC components usually supplied by Microsoft. Only install them if your machine is not already equipped with them.
- ODBC Samples: examples of use with Word, Excel and Visual Basic delivered with **Tun SQL**.
- Help Files: **Tun SQL** on-line help.

NIS and Print Applications

This component includes three subcomponents:

- NIS Application: Network Information Service for centralized management of the network's resources.
- Print Applications: LPR printer redirection and LPD printer sharing.
- Help Files: On-line help for NIS, printer sharing and redirection.

For further information on the use of NIS, please refer to the user manual **TCP/IP Network Services**

Win 3.X

TCP/IP Stack (16-bit version only)

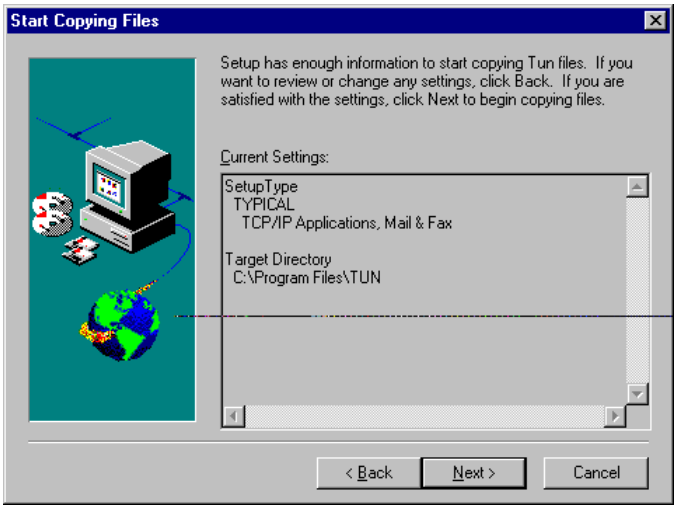
This component includes the following three subcomponents:

- TCP/IP Stack: DLLs et VxD drivers for TCP/IP stack.
- Network Card Driver: Packet Driver, NDIS Driver.
- Help Files: on-line help for the TCP/IP stack.

For more information on this component, please consult the manual **Tun KERNEL**.

Then click the button **Next**.

5. The following window will appear:

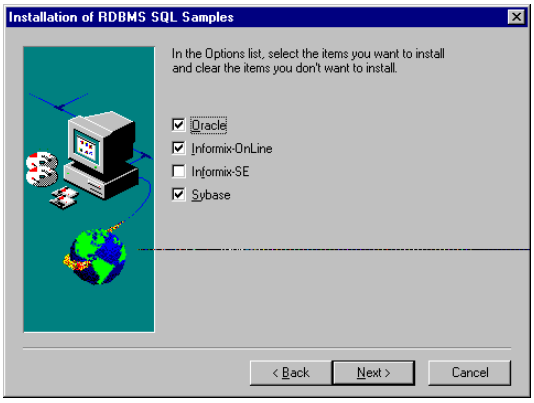


Then click the button **Next** if you are satisfied with the installation options.

Win 3.X

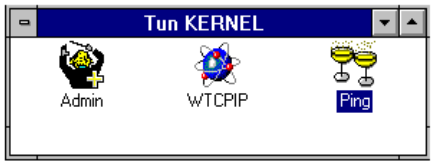
6. At the end of the installation process, if you requested the installation of the TCP/IP stacks, a window will appear proposing the immediate configuration of **Tun KERNEL**. If you agree to this option, the installation procedure switches immediately to the **Tun KERNEL** configuration screens as described in the chapter **Using and Configuring Tun KERNEL** in the **Tun KERNEL** manual. If you refuse this option, it will be possible to return to it later by clicking on the **Admin** icon in the **Tun KERNEL** or **Tun NET** groups under the Windows Program Manager.

If you requested the installation of the examples, a new dialog box will be displayed. Select the DBMS (Oracle, Informix On-Line, Informix SE, Sybase) for which you wish to install the examples:

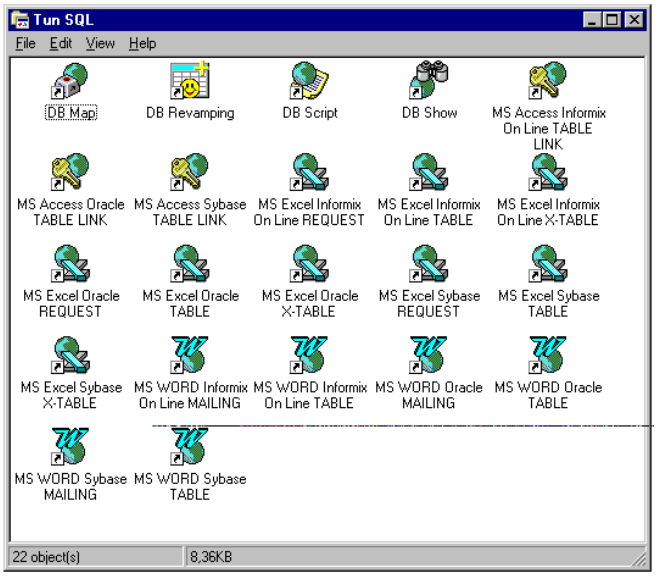


Note:	All the examples will be transferred to the hard disk. Only the icons for the chosen examples are created.

7. The installation process has now finished. For an installation under Windows 3.x, you should be able to see the new **Tun KERNEL** group with the following icons (if you chose to install it):



and also the new **Tun SQL** group with these icons:



Note:	If the **Tun NET** software is already installed on the PC, the **Tun KERNEL** icons will be included in the **Tun NET** group under Windows Program Manager.

Tun SQL CONFIGURATION

Tun SQL may be supervised and configured under Windows using the application **Tun DB Show** in the **Tun SQL** group or by clicking on the icon **ODBC** in the Windows Control Panel.

DIRECTORY STRUCTURE AND INSTALLED FILES

The installation procedure installs the following files in the \TUN\SQL default directory:

BIN\	Executable File Directory
BIN**TUNODBC.DLL**	Tun SQL ODBC Driver (Windows 16 bits)
BIN**TUNODB32.DLL**	Tun SQL ODBC Driver (Windows 32 bits)
BIN**TUNTRANS.DLL**	Tun SQL Translator (Windows 16 bits)
BIN**TUNTRS32.DLL**	Tun SQL Translator (Windows 32 bits)
BIN**TUNDB.DLL**	Application DLL (Windows 16 bits)
BIN**TUNDB232.DLL**	Application DLL (Windows 32 bits)
BIN**TUNGEN.DLL**	Application DLL (Windows 16 bits)
BIN**TUNGEN32.DLL**	Application DLL (Windows 32 bits)
BIN**TUNMAP.DLL**	Tun SQL Virtual ODBC Driver (Windows 16 bits)
BIN**TUNMAP32.DLL**	Tun SQL Virtual ODBC Driver (Windows 32 bits)
BIN**DBSHOW.EXE**	Tun SQL Server Explorer (Windows 16 bits)
BIN**DBSHOW32.EXE**	Tun SQL Server Explorer (Windows 32 bits)
BIN**DBMAP.EXE**	Translation Table Editor (Windows 16 bits)
BIN**DBMAP32.EXE**	Translation Table Editor (Windows 16 bits)
BIN**DBSCRIPT.EXE**	SQL Script Interpreter (Windows 16 bits)
BIN**DBSCRIPT32.EXE**	SQL Script Interpreter (Windows 32 bits)
BIN**DBREVAMP.EXE**	Database Revamping Tool (Windows 16 bits)
BIN**DBREVA32.EXE**	Database Revamping Tool (Windows 32 bits)

ETC\	Parameter Files Directory
ETC***.INI**	Initialization files (1 per DLL/EXE)
ETC***.LGFR, *.US, ...**	Language Files
ETC***.HLP**	Help Files
ETC***.TTT**	Translation Tables

DEMO\	Demonstration Directory
DEMO\DB	Demo Database Directory
DEMO\DB**XXXCREAT.SQL**	SQL Script for DEMO DB Creation
DEMO\DB***.BMP**	Images of European Country Maps
DEMO\WORD\	Word Demo Directory
DEMO\WORD***.DOC**	2 Files
DEMO\EXCEL\	Excel Demo Directory
DEMO\EXCEL***.XLS**	3 Sample Files
DEMO\ACCESS	Access Demo Directory
DEMO\ACCESS**XXXDEMO.MDB**	Sample Access Database with Remote Table Link
DEMO\VISUALBA\	Visual Basic Demo Directory
DEMO\VISUALBA**TUNVBSQL.FRM**	Visual Basic Source Code
DEMO\VISUALBA**TUNVBSQL.MAK**	Visual Basic Makefile
DEMO\VISUALBA**TUNVBSQL.EXE**	Visual Basic Exe File

CHAPTER 3 - INSTALLATION UNDER UNIX

ACQUIRING Tun SQL UNIX COMPONENTS

As explained in the introduction, the Tun SQL UNIX components have to be installed on a server before the software can be used. There are as many Tun SQL servers as there are machines and DBMSs supported.

At the moment the following platforms and DBMSs are supported:

	Oracle 7	Informix 5	Informix 7
SCO UNIX 3.2.x v4.2	Yes (7.0)	Yes	
SCO UNIX 3.2.x v5.0	Yes (7.0/7.3)	Yes	Yes
SunOS 4.1.3	Yes (7.1)	Yes	
Solaris 2.5	Yes (7.2)	Yes	
AIX 3.2	Yes (7.0)	Yes	
AIX 4.1			Yes
HP-UX 9.x	Yes (7.2)	Yes	
HP-UX 10.x	Yes (7.2)	Yes	
OSF1 v3.2	Yes (7.2)	Yes	

	Sybase 10	DB2	Progress 6
SCO UNIX 3.2.x v4.2	Yes		Yes
SCO UNIX 3.2.x v5.0	Yes		Yes
SunOS 4.1.3	Yes		Yes
Solaris 2.5	Yes		
AIX 3.2	Yes		
AIX 4.1		Yes	Yes
HP-UX 9.x	Yes		Yes
HP-UX 10.x	Yes		Yes
OSF1 v3.2	Yes		Yes

Note: This list is not exhaustive and ESKER is continuing to port its servers to other hardware platforms. If your machine is not in the list, please consult your distributor to see if the porting has not already been done.

Given the volume occupied by each of these servers (about 1 Mb), it has not been possible to include them on the **Tun SQL** installation disks. Only the CD-ROM version of **Tun PLUS** includes them in the directory **\RDBMS**.

If you acquired **Tun PLUS** or **Tun SQL** on disks, you have to fill in the reply coupon included in the package to obtain the UNIX server or servers which suit your configuration free of charge.

PRELIMINARY CHECKS

The following conditions have to be fulfilled for the **Tun SQL** server you wish to install to run:

- The UNIX machine you wish to use is connected to a network and uses the TCP/IP protocol.
- The UNIX machine has one of the following DBMSs operating: Oracle, Informix or Sybase.
- You have a disk containing the **Tun SQL** server for the machine and DBMS in question.

LOADING AND CONFIGURING A Tun SQL SERVER

Each **Tun SQL** UNIX server is delivered on a separate disk from the Windows part of the software. The disk contains only one server which is a file with the extension .TAR and which consists of the following:

- A compiled and compressed version of the **Tun SQL** UNIX server for a particular machine and DBMS (tunodbc.ora.Z, tunodbc.ifx.Z...).
- A README file.
- An installation procedure (tunsql.install).
- An installation program (installsh).
- Installation settings (file with the extension .ins).

To install a **Tun SQL** UNIX server, follow these steps while logged in as **root** on the UNIX machine:

1. Transfer the file XXX.TAR, where XXX refers to the type of DBMS used (Oracle, Informix 5, Informix 7, Sybase, DB2, Progress), from the disk or CD-ROM **in binary mode** to the directory **/tmp** on the UNIX machine (using **Tun FTP** from **Tun NET** for example).
2. Enter the following command to go to the directory **/tmp** on the UNIX machine:

```
cd /tmp
```

3. Expand the file **tunsql.tar** with the command:

```
tar xvf tunsql.tar
```

After installing the necessary files for the Tun SQL server, it has to be configured. The purpose of the configuration is to inform the **Tun SQL** UNIX server of the installation characteristics of the DBMS it has to interface with:

4. Run the shell script **tunsql.install**.

5. As soon as it is run, the installation procedure displays the type of machine and DBMS the **Tun SQL** UNIX server is designed for. Check that the displayed settings suit your configuration.

The procedure then requests confirmation of the installation and asks for the following information.

Installation directory

As the name implies, this is the directory in which the executable programs and the **Tun SQL** UNIX configuration files are installed. The procedure proposes the default directory **/usr/tunsql** but this can be changed.

The installation parameters of the DBMS

Depending on the type of DBMS used, the installation procedure will ask for specific settings such as the name of the DBMS installation directory or the default database. Exact answers are required. If certain system variables have been assigned values, the installation procedure will try and read them so as to be able to help you in this operation.

Updating the files *services* and *inetd.conf*

Since the **Tun SQL** UNIX server is a TCP/IP server, it must be recognized by the super daemon **inetd**. Should you require it, this operation can be carried out automatically by the installation procedure. By default, the procedure uses the TCP/IP numbers from 5370 to 5380. They may be changed on condition that the change is recorded in the SERVICES file of the TCP/IP stacks on the PC.
The installation procedure sends the signal SIGHUP to the program **inetd** so it can take account of the changes made to the file **/etc/inetd.conf** with the following command:

```
kill -1 no_pid_inetd
```

If there have been no problems with the installation, the **Tun SQL** UNIX server is now ready to use.

INSTALLATION OF ANOTHER Tun SQL SERVER ON THE SAME MACHINE

If several DBMSs share the same machine, the same number of **Tun SQL** UNIX servers can be installed. To do this, carry out the same operations with the disk containing the required server.

INSTALLED AND UPDATED FILES

Tun SQL files

The following files are created in the installation directory after execution of the command **tunsql.install**:

File	Function
tunodbc200.*	**Tun SQL** UNIX server program (one per DBMS)
param.*	Server startup parameters (one configuration per DBMS)
config.*	Working and security parameters (one configuration per DBMS)

System files

The following system files are updated by the installation script **tunsql.install**:

/etc/services
/etc/inetd.conf

Both the files are saved before the changes under the names **/etc/services.bak** and **/etc/inetd.conf.bak**. If the **tunsql** service does not exist, the following line or lines will be added to the file **/etc/services**:

```
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
```

There should be the same number of lines as there are **Tun SQL** servers installed on the UNIX machine.

If the service **tunsql** has not already been made active, the following line or lines will be added to the file **/etc/inetd.conf**:

```
tunodbc200.xxx stream tcp nowait root /usr/tunsql/tunodbc200.xxx
tunodbc200.xxx -f=/usr/tunsql/param.xxx
```

for each of the **Tun SQL** servers installed on the UNIX machine, where xxx stands for the relative file extension as follows:

- ora for Oracle
- ifx for Informix
- syb for Sybase
- db2 for DB2
- pro for Progress

There should be as many lines as there are **Tun SQL** servers installed on the UNIX machine.

CHECKING THE INSTALLATION

Execute the following command to check that the installation of the **Tun SQL** UNIX server has been correctly carried out:

```
netstat -a | grep tunodbc200
```

If all is working correctly, the command should provoke the display of the following line:

```
tcp  0  0 *.tunodbc200.   *.*        LISTEN
```

In fact, there should be as many lines similar to this one as there are **Tun SQL** UNIX servers installed on the machine. If, on the contrary, there is no reply or not enough lines, this means the installation has not been carried out correctly. You then have to check the contents of the files **/etc/services** and **/etc/inetd.conf** and send the signal **1** to the **inetd** process or stop and restart TCP/IP.

STARTING THE Tun SQL UNIX SERVER

If the **Tun SQL** UNIX servers have been declared to the program **inetd** by means of the file **/etc/inetd.conf**, they will start automatically as soon as a Windows application needs to use them (**Tun DB Show**, for example).

Note:	The **Tun SQL** UNIX servers will not start if the **inetd** process is not running on the UNIX machine.

Checking the Tun SQL UNIX server

To check that the installed **Tun SQL** UNIX servers are working properly, run the **Tun DB Show** application under Windows as described in the next chapter.

Working parameters of the Tun SQL UNIX server

The access rights of particular PCs or users can be limited to specific databases. This is carried out by writing to the **config.xxx** files in the directory **etc** of the **Tun SQL** UNIX server installation directory (Cf. **Reference Guide**).

PART 2
CLIENT/SERVER COMMUNICATION

CHAPTER 4 - CONFIGURATION AND USE

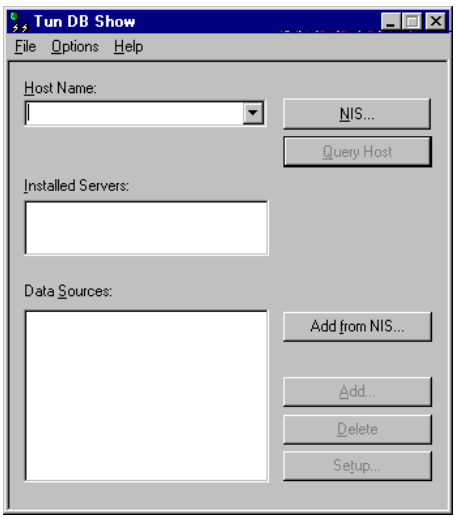
VERIFYING THE FUNCTIONING OF Tun SQL

Executing Tun DB Show

After the installation and configuration of **Tun SQL** under Windows and under UNIX, it is necessary to verify it is functioning correctly. This can be done by running the application **Tun DB Show**.

Run the program by clicking on the **Tun DB Show** icon in the **Tun SQL** group under Windows 3.x or from the Windows 95 Start menu.

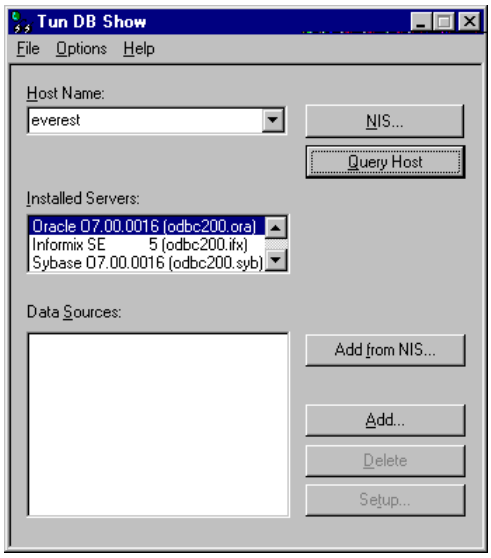
The following window is displayed:



This utility can be used to query a UNIX host on the network to see if one or more **Tun SQL** UNIX servers are present. Enter the name or IP address of the host in the field **Host Name** and then press the button **Query Host**.

If the application **Tun NIS** has been installed on the PC and the network administrator has configured the NIS tables, the **NIS** button can be used to access the servers installed on the network. Consult the manual "**TCP/IP Network Services**" for information on configuring **Tun NIS**.

If one or more **Tun SQL** UNIX servers are correctly installed on the remote machine, the list **Installed Servers** should appear as follows (at least one line):



Each line contains the following information:

- The name of the DBMS which the**Tun SQL** UNIX server is interfaced with.
- The version number of the DBMS.
- The name of the UNIX executable file which functions as the server (tunodbc200.ora, for example).

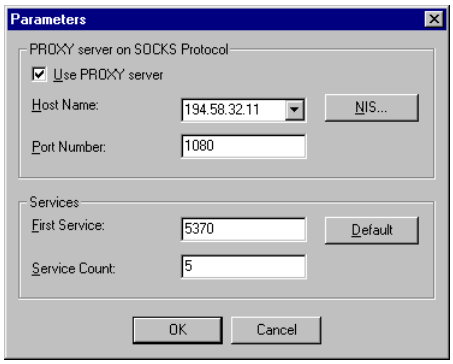
Note:	With some DBMSs (Informix On-Line, for example), selecting the server in the list displays the list of databases managed by the DBMS in the appropriate column (Databases).

If there are no **Tun SQL** servers in the list, it means that a problem occurred during installation. In this case all the operations and checks described in the previous chapters have to be carried out again.

Parameters

It is possible to specify network security measures in the **Tun DB Show Options➔Parameters** menu. Namely, you can:

- Enter the settings for a Proxy server: access to an outside server will then go through a gateway machine of the Proxy type. The settings entered in **Tun DB Show** will be viable for all the Tun applications on the machine that allow access to servers outside the local network. Select the check box **Use PROXY Server** and enter the name or the IP address of the Proxy server and the number of the port to be used.
- Manage the services: a service number from 5370 through 5374 is associated with each **Tun SQL** server process. The correcting functioning of **Tun SQL** requires the definition of a First Service and the number of services possible; this enables it to detect the different database systems accessible to a **Tun SQL** server. The default **First Service** is 5370 and the **Service Count** default is 5.



CREATING A SAMPLE DATABASE

To use the examples supplied with the **Tun SQL** software package, a specific database must be created in one of the available DBMSs. This can be done using the tools belonging to the DBMS in question.

Given the difficulty inherent in creating a database in some DBMSs (Oracle), an existing database may be used provided that it does not contain sensitive data. It is preferable to call the newly created database **tunsqldemo** for a better understanding of the next part of the documentation.

CREATING A DATA SOURCE

Introduction to data sources

So that a particular driver and database can be used, ODBC-compliant applications have to recognize a data source.

Data source is a key term which refers to the name of the ODBC driver used (for example, **tunod32.dll**) and the information required to make it function. This information is as follows:

- The name or the IP address of the remote UNIX host.
- The type of DBMS (Oracle, Informix, Sybase, DB2, Progress).
- The name of the database.
- Comments.
- Optional supplementary information.

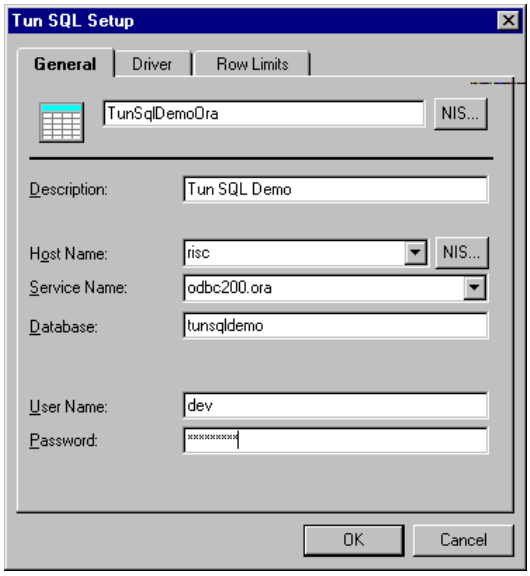
Creating a data source

All the examples supplied with the **Tun SQL** package use the same data source. Run the application **Tun DB Show** again and carry out the following operations to create this data source:

- Enter the name or the IP address of the host on which the database has been installed.
- Click on the button **Query Host** to display the list of **Tun SQL** UNIX servers.
- Select the server associated with the DBMS containing the sample database.
- Click on the button **Add**.

The button **Add From NIS...** allows access to the list of data sources available on the network if the application **Tun NIS** has been installed and the administrator has configured the NIS tables. Consult the manual "**TCP/IP Network Services**" for the configuration of **Tun NIS**.

Clicking on the button **Add** opens this dialog box.



General tab

The different fields on the first tab are as follows:

Data Source Name

The icon  represents the field containing the name of the data source as used by the ODBC-compliant applications. To be able to connect to the sample database, it is imperative that the data source is called:

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

The button **NIS...** allows access to the list of data sources available on the network if the application **Tun NIS** has been installed and the administrator has configured the NIS tables. Consult the manual "**TCP/IP Network Services**" for the configuration of **Tun NIS**.

Description

This field contains a comment associated with the data source.

Host Name

This name contains the IP address or the name of the host on which the database is installed that the user wishes to use.

Service Name

This field contains the name of the **Tun SQL** server process associated with the DBMS in which the required database has been created (for example, **tunodbc200.ora**).

If you are using different TCP/IP stack from **Tun KERNEL**, you should complete the file **services** in the TCP/IP software used with the following values:

```
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
```

Database

This field contains the name of the database that the user wishes to use (**tunsqldemo** in this case for the sample database).

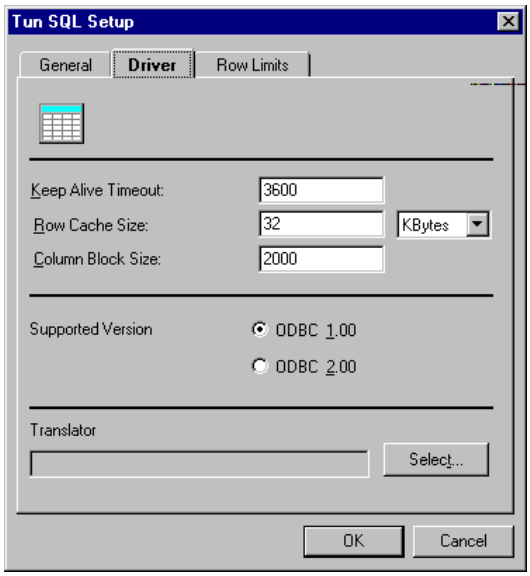
User Name

This field contains the name of a user authorized to access the database.

Password

Enter the password associated with the user.

Click the **Driver** tab to display the ODBC driver configuration window:



The tab contains the following fields:

Keep Alive Timeout

Since the PC is a machine which is subject to frequent software and hardware failures, the **Tun SQL** UNIX server has to check regularly that the PC is still under power. To this effect, it regularly sends packets to the PC. If it does not reply in a time of **n** seconds the process stops. This field holds the timeout value (the default is 1 hour).

Row Cache Size

Indicates the size of the packets of data extracted from a table during an SQL "select" operation. The value may be expressed in kilobytes or number of lines.

If the value equals 1 there is one TCP packet per row retrieved. If the value equals 100 then the rows will be grouped in packets of 100 up to the number of rows actually retrieved. This value makes it possible to optimize the exchanges on the network. The optimum value lies between 50 and 150. The default value is 32 Kb.

Column Block Size

Indicates the fragmentation unit to be used when very wide columns have to be retrieved from the database (large amounts of text or images). If this value is too low it means the number of exchanges on the network will be considerably increased.

Supported version

At the moment there are two versions of the ODBC API, numbered 1.00 and 2.00. Some applications are only compatible with ODBC version 1.00 (Microsoft's "Access") and do not work with drivers from a higher version. The **Tun SQL** ODBC driver, which is compatible with version 2.00, is able to emulate version 1.00 so that these applications can nevertheless run on it.

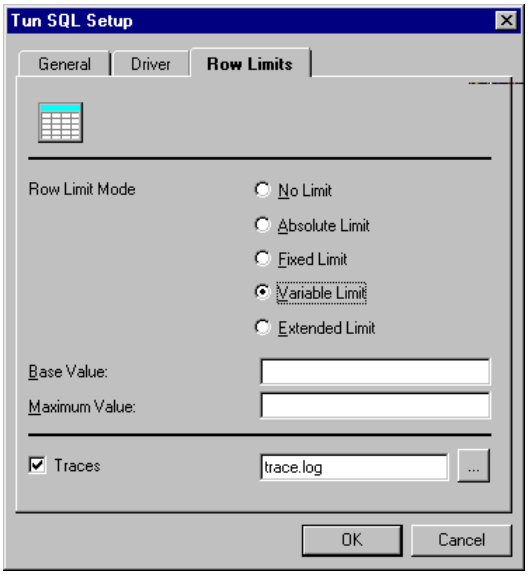
Select the appropriate check box to indicate to the ODBC driver the level of compatibility required for a given application.

Translator

Given the differences in the notation of accented characters in the Windows (CP850) and UNIX (ISO8859) environments, it is sometimes necessary to set up character conversion tables for the ODBC driver. The **Select** button enables the user to choose the required conversion table. In the sample database this field may be ignored since all the texts it contains are in English.

Note:	The conversion tables can be created or edited using the application **Tun DB Map**.

Click on the tab **Row Limits** to display the limits dialog box:



Row Limit Mode

Some office applications allow the user to compose his own SQL requests. In other cases, certain applications tend to read the contents of an entire table before displaying it on the screen. This does not pose any particular problems since they are dealing with small tables in a local database.

On the other hand, insurmountable problems may arise when it is a question of very large tables in a centralized, remote database. In this case, there is a considerable number of exchanges on the network and there may be insufficient memory in the PC to store the received data. The PC has to be frequently rebooted after such a "select" query.

To compensate for this problem, the **Tun SQL** ODBC driver incorporates the notion of limits which can be defined on the **Row Limits** tab.

Five types of limits are recognized by the **Tun SQL** ODBC driver:

Limit	No limit is imposed by the ODBC driver
solute Limit	The ODBC driver will refuse to load more than **n** rows during one **select** request. No messages will be displayed to inform the user.
ed Limit	The ODBC driver will refuse to load more than **n** rows during one **select** request. A message will be displayed informing the user.
riable Limit	The ODBC driver will refuse to load more than **n** rows during one **select** request. It will however display a message proposing to load more within the limit of the **Maximum Value**.
tended Limit	The ODBC driver will refuse to load more than **n** rows during one **select** request. It will however display a message to load more without a Maximum Value. In this case the message displayed is really only a warning.

Traces

You can select the check box **Traces** to save the trace of your SQL queries in a **.log** file which can later be consulted. This option lets you see what the ODBC driver does when you pass it an SQL query.

Click the button **...** to choose the directory in which you wish to place this file.

> **Notes**: In this chapter you were asked to create a data source called **tunsqldemoXXX** which all the examples supplied with the **Tun SQL** software package refer to.
>
> If you wish to use **Tun SQL** with other applications and databases, the necessary data source must be created each time. As a general rule, there must be a particular data source for each application and for each database used.
>
> A data source may be created directly using the ODBC application in the **Control Panel**.

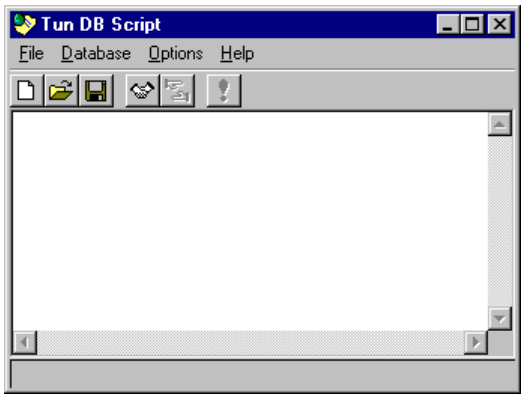
TRANSFERRING THE DEMONSTRATION DATABASE

The **Tun SQL** package is delivered with a sample database to be used with the examples included. This database must be downloaded from the PC to the data source **tunsqldemoXXX** which you were asked to create in the preceding sections.

> **Note**: The variable XXX in the name of the data source may take one of the following values:
>
> - **Ifx** For Informix On-Line
> - **Ise** For Informix SE
> - **Ora** For Oracle
> - **Syb** For Sybase

To perform the downloading, run the program by clicking on the **Tun DB Script** icon in the **Tun SQL** group under Windows 3.x or from the Windows 95 Start menu.

When this application starts running, the following window appears:



Loading the SQL batch file to create the database

The database of your choice can be loaded using the option **File➔Open** or by clicking on the button  .

To download the sample database, it is necessary to load the file **DEMO\DB\XXXCREAT.SQL** from the installation directory of **Tun SQL**.

Connection with the data source

Before you can run the SQL batch file, it is necessary to establish a connection with a data source. To do this, click on the button  or use the option **Database➔Connect**.

This operation displays a dialog box asking for the name of the data source to use. If the connection can be established, the batch file will run.

To download the sample database, select the data source **TunSqlDemoXXX** which was defined earlier.

Execution

To execute the SQL batch file, select the option **Database➔Execute** in the main menu or click on the button  . **Tun DB Script** will then submit the SQL commands consecutively to the database which corresponds to the selected data source. If there is an error, **Tun DB Script** will stop running and display an error message.

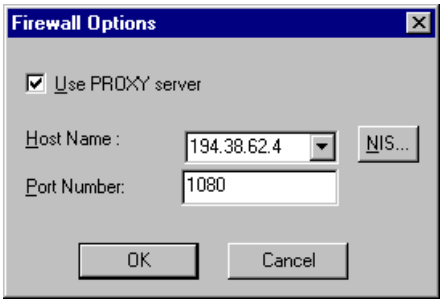
Disconnection of the data source

After execution, the data source has to be disconnected by clicking on the button  or by selecting the option **Database➔Disconnect** in the main menu. A disconnection of the data source is also provoked on quitting the application.

Note:	Although the principal purpose of **Tun DB Script** is to download the **Tun SQL** sample database, it also has other uses. In fact, **Tun DB Script** can execute whole lists of SQL commands to create other databases or to update or completely clear extremely large tables.

Use of a Proxy server

Tun DB Script features a security option **Option➔Firewall** with which you can elect to use a Proxy server: access to an outside server will then pass through a gateway machine of the Proxy type.



The settings entered in **Tun DB Script** will be viable for all the Tun applications on the PC that enable access to servers outside the local network. Simply select the check box **Use PROXY Server** and enter the name or the IP address of the Proxy server and the number of the port to be used.

CHARACTER CONVERSION TABLES

This section may be omitted on the first reading.

Difference in notation between the different computing systems

Although all the characters used in English have been perfectly codified in the ASCII table (0 to 127), the same is not the case for the special characters or accents used by other languages (e.g. French, German, Spanish, Italian, etc). Despite the fact that norms exist (e.g. ISO 8859), they are not always implemented in every computing system.

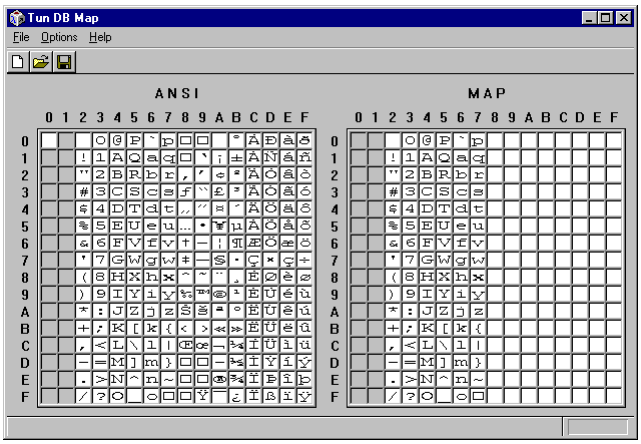
In so far as **Tun SQL** enables a computing system (the PC) to access data located in another computing system (a DBMS operating under UNIX), it has been thought appropriate to include in **Tun SQL** a mechanism to handle the differences in notation between the national characters in different systems. This mechanism enables the PC to display an accented character (an **é** for example) even if it has been coded differently in the DBMS under UNIX.

Creation of conversion tables

The mechanism supplied with **Tun SQL** uses "conversion tables" which can be created or updated with the application **Tun DB Map**.

Run the program by clicking on the **Tun DB Map** icon in the **Tun SQL** group under Windows 3.x or from the Windows 95 Start menu.

The following screen is displayed:



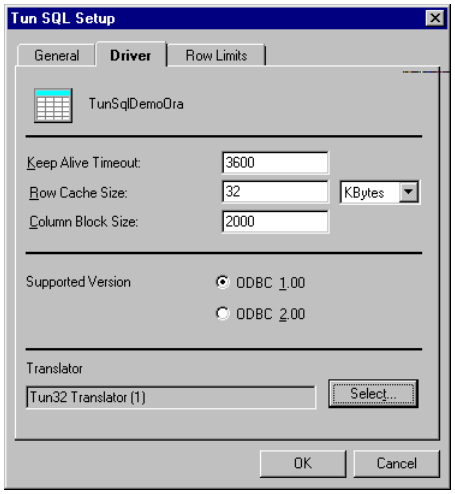
The table on the left shows all the characters available on the PC (ASCII and extended ASCII). The table on the right should have the same characters but in positions which correspond to the notation used by the DBMS on the UNIX machine. The first 128 positions in the table on the right are already filled since they are the same in both systems.

To assign a character to one of the positions in the right-hand table, select a character in the left-hand table and "drag" it across to one of the squares in the right-hand table by keeping the mouse button pressed.

It is possible to create as many conversion tables as are necessary using the **File** option in the general menu and saving the files with the extension **.ttt**.

Implementation of conversion tables

For conversion tables to be taken into account, they have to be associated with a data source using the dialog box displayed by **Tun DB Show** to this effect: enter the details in the section **Translator**:



or press select to display the following dialog box:

Select which ODBC translator you wish to use and click **OK**.

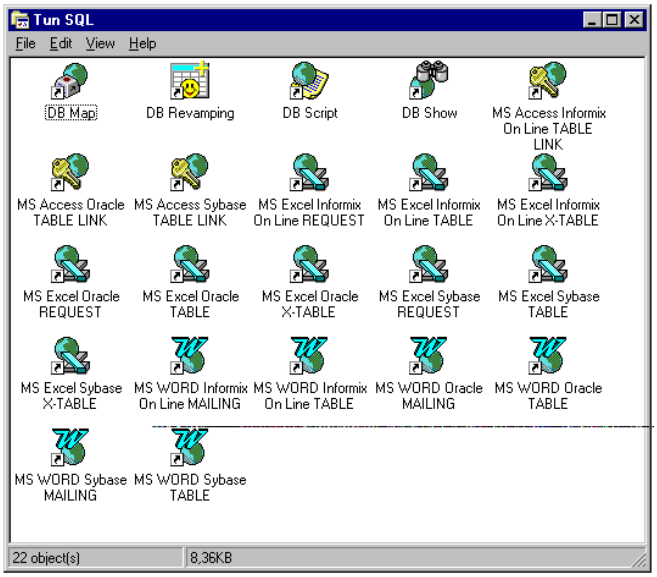
CHAPTER 5- Tun SQL AND OFFICE APPLICATIONS

A group of programs was created on your PC during the installation of **Tun SQL** which consists of icons referring to examples of the use of **Tun SQL** with the Microsoft programs Excel, Word, Access and Visual Basic.

It is only possible to use the examples supplied with **Tun SQL** if a data source entitled **TunSqlDemoXXX** has been created for the sample database delivered with the software. The database **tunsqldemoXXX** must also have been created and initialized using **Tun DB Script** and the file XXX**CREATE.SQL**. If these operations have not been carried out, refer to the previous chapter which explains the necessary procedure, step by step.

Note:	The variable **XXX** in the data source name must be replaced with one of the following values, depending on the database used: • **Ifx** For Informix On-Line • **Ise** For Informix SE • **Ora** For Oracle • **Syb** For Sybase

The figure below shows an example of a **Tun SQL** program group. The contents of the group depend on the options chosen during installation. The example includes all the possible options:



USING Tun SQL WITH MICROSOFT EXCEL

The following example uses an **Oracle** database. It can be displayed by clicking on the icon MS-Excel Oracle TABLE.

Choose the icon which corresponds to your database (e.g. MS-Excel Informix TABLE).

Tun SQL can be used with Excel to **Read External Data**, for example, or to **Update Data** from the DBMS.

In this example, Excel opens a spreadsheet and the graph associated with it. The spreadsheet contains only zeros.

	90	91	92	93
Germany	0	0	0	0
Belgium	0	0	0	0
Denmark	0	0	0	0
Spain	0	0	0	0
France	0	0	0	0
United Kingdom	0	0	0	0
Greece	0	0	0	0
Ireland	0	0	0	0
Italy	0	0	0	0
Luxembourg	0	0	0	0
The Netherlands	0	0	0	0
Portugal	0	0	0	0

To update these values with those in the database, select (click with the mouse) the values to update and then select the main menu option **Data➔Update Data**.

Excel then runs Microsoft's **Microsoft Query** in order to use the appropriate queries to fetch the required data from the database.

MS Query uses the data source associated with the example, which must, of course, exist.

Connection to the host is established when the login and password have been entered.

> **Note**: To render the connection procedure automatic and not have to supply login identification and a password each time, add the following two lines to the section relating to the data source, which is to be used in the file C:\WINDOWS\ODBC.INI:
>
> **UID=your_login_ID**
> **PWD=your_password**

The selected data and the associated graph will then be updated.

	90	91	92	93
Germany	75	75	77	0
Belgium	75	76	76	0
Denmark	75	75	75	0
Spain	77	77	76	0
France	77	77	77	0
United Kingdom	76	76	76	0
Greece	77	77	77	0
Ireland	74	74	75	0
Italy	77	76	78	0
Luxembourg	75	75	75	0
The Netherlands	77	77	77	0
Portugal	75	75	75	0

Creation of an Excel file from a database

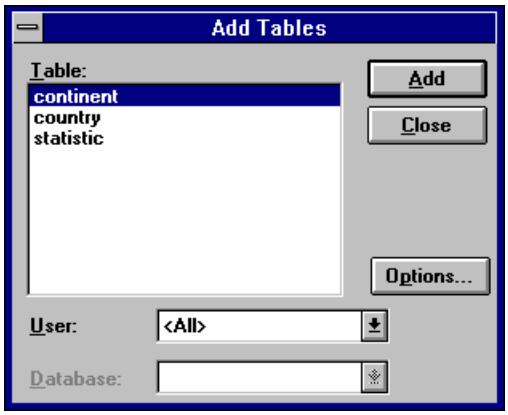
We are now going to create another table with our own SQL queries using the same database.

The objective is to show the progress of inflation in the different member states of the European Community since 1990.

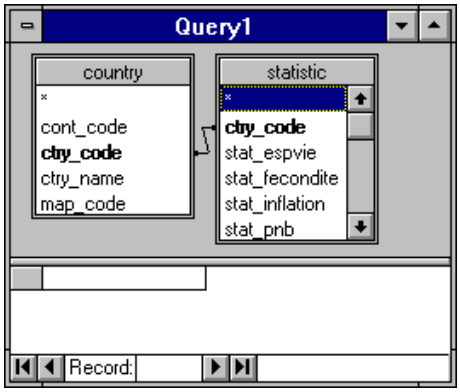
To do this, a new Excel file must first of all be created and the spreadsheet prepared to receive the data; e.g. create columns for the years 1990, 1991, 1992 and 1993. The preparation depends on the final form you

wish to give your document. Next, select the option **Data➔Read External Data** in the main menu which will execute **MS Query**.

Select the data source to use (in this case, **TunSqlDemoXXX**). When you have been connected to the server, **Microsoft Query** requires you to select the tables for which you wish to create your queries.



When the two tables have been selected (country, statistic), they appear in the MS-Query window with the different keys which link them.

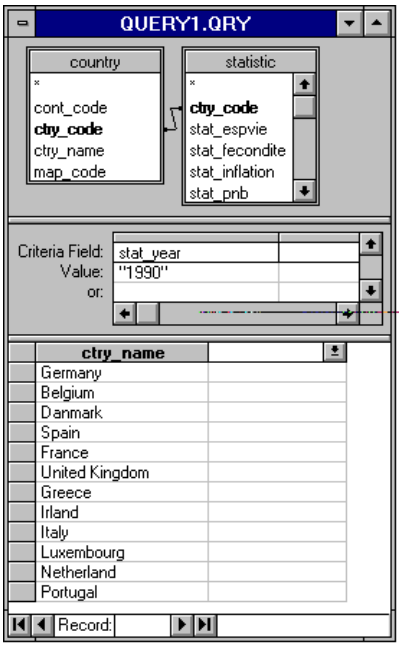


We are then able to generate our own SQL queries to access the required data. To do this we have to complete the following fields:

- **ctry_name**
- **stat_inflation**
- **stat_year**

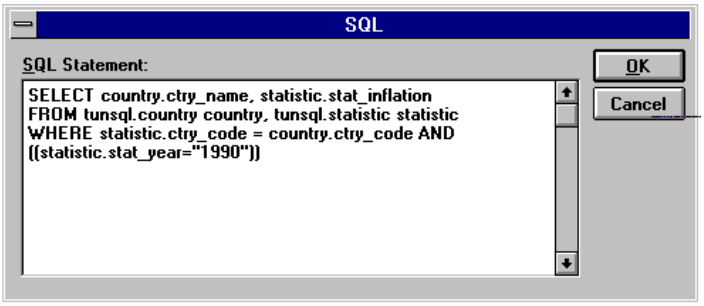
The field **stat_year** will be used to designate the **condition** for our SQL query.

Select the field **ctry_name** and "drag and drop" it into the active cell. The country names will then each be selected four times (for each year). A condition relating to the year can be added to the query to avoid this repetition. To achieve this, select the option **Display➔Criteria** in the main menu and enter the selection criterion 1990 in the dialog box displayed for this purpose.



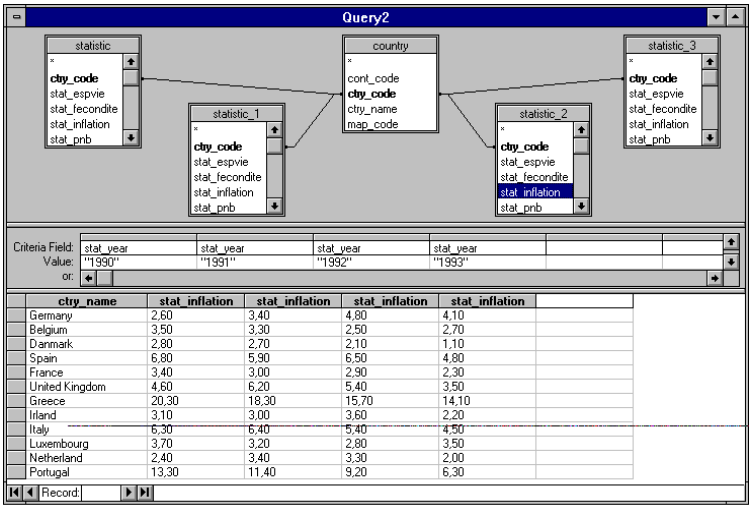
The second part of the query relates to the field **stat_inflation**. For this field to be taken into account, proceed as for **ctry_name** and "drag and drop" it into the next available active cell.

It is then possible to view the syntax of the SQL query by selecting the option **Display➔SQL** in the general menu. If necessary, the syntax can be modified.



The extracted data can thus be sent to Excel with the option **File➔Transmit Data to Microsoft Excel** (keeping or not keeping the column titles).

You may then repeat the operation for the following years (1991, 1992, 1993). It is possible to make only one query for the four years by opening the same table in MS Query several times, as in the example below.



ctry_name	stat_inflation	stat_inflation	stat_inflation	stat_inflation
Germany	2,60	3,40	4,80	4,10
Belgium	3,50	3,30	2,50	2,70
Danmark	2,80	2,70	2,10	1,10
Spain	6,80	5,90	6,50	4,80
France	3,40	3,00	2,90	2,30
United Kingdom	4,60	6,20	5,40	3,50
Greece	20,30	18,30	15,70	14,10
Irland	3,10	3,00	3,60	2,20
Italy	6,30	6,40	5,40	4,50
Luxembourg	3,70	3,20	2,80	3,50
Netherland	2,40	3,40	3,30	2,00
Portugal	13,30	11,40	9,20	6,30

The following SQL query will then be used:

```
SELECT country.ctry_name,
statistic.stat_inflation,
statistic_1.stat_inflation, statistic_2.stat_inflation,
statistic_3.stat_inflation
FROM tunsql.country country, tunsql.statistic statistic,
tunsql.statistic statistic_1, tunsql.statistic statistic_2,
tunsql.statistic statistic_3
WHERE statistic.ctry_code = country.ctry_code AND
statistic_1.ctry_code = country.ctry_code AND
statistic_3.ctry_code = country.ctry_code AND
statistic_2.ctry_code = country.ctry_code AND
((statistic.stat_year="1990") AND
(statistic_1.stat_year="1991")
AND (statistic_2.stat_year="1992") AND
   (statistic_3.stat_year="1993"))
```

The results can be organized using the different tools in MS Query in order to return a completed table to Excel.

Country	1990	1991	1992	1993
Germany	2,6	3,4	4,8	4,1
Belgium	3,5	3,3	2,5	2,7
Denmark	2,8	2,7	2,1	1,1
Spain	6,8	5,9	6,5	4,8
France	3,4	3	2,9	2,3
United Kingdom	4,6	6,2	5,4	3,5
Greece	20,3	18,3	15,7	14,1
Ireland	3,1	3	3,6	2,2
Italy	6,3	6,4	5,4	4,5
Luxembourg	3,7	3,2	2,8	3,5
Netherlands	2,4	3,4	3,3	2
Portugal	13,3	11,4	9,2	6,3

A number of examples are supplied with **Tun SQL** which use different Excel functions. Study them carefully to familiarize yourself with the use of these programs.

USING Tun SQL WITH MICROSOFT WORD

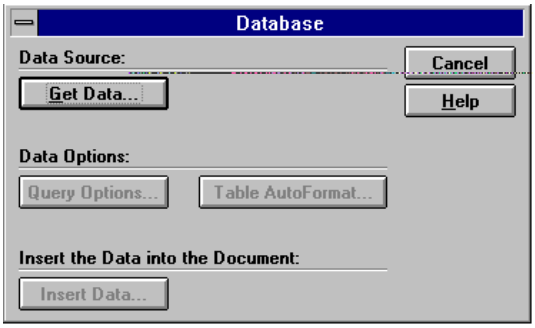
Tun SQL can also be used with Microsoft **Word**. We are going to create two examples similar to those supplied with **Tun SQL.**

- Creation of a table showing population changes in the different member states of the European Community between 1990 and 1992.
- Use of Word to produce a mail merge.

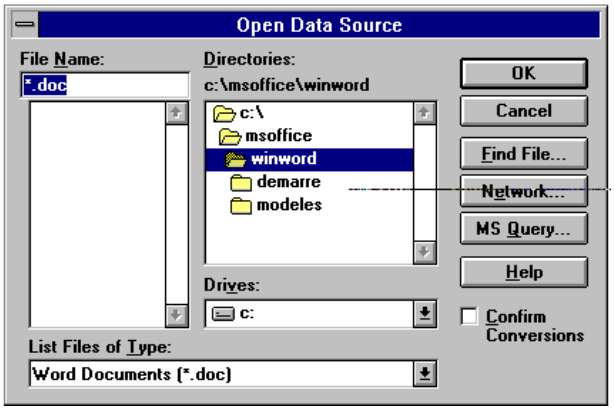
Creation of a table using Word and Tun SQL

First of all, open a new Word document. The objective is to retrieve data stored in the DBMS and put this data in a table.

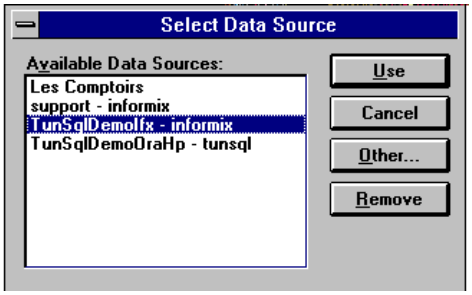
Select the option **Insertion➔Database** in the main menu.



Click on the button **Get Data**. This opens a window in which you have to select the application **MS Query** in order to query the database:



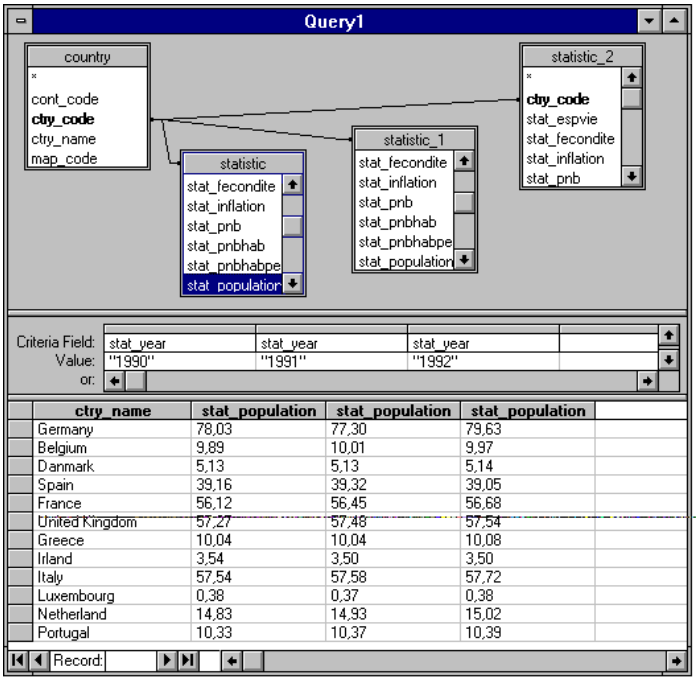
The procedure is the same as that described for Excel. Select the data source **TunSqlDemo**



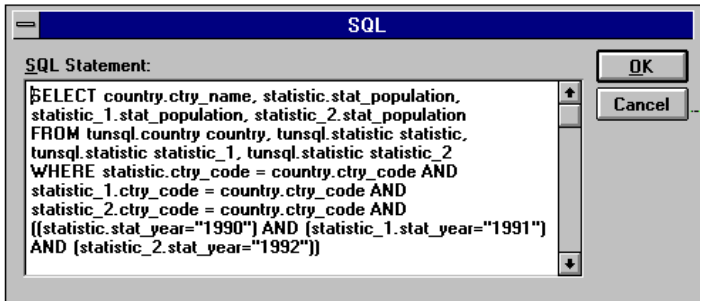
When **MS Query** has initiated the connection, enter the login details and the password to establish the connection with the database.

Note:	To automate the login procedure, modify the file ODBC.INI as described for the Excel example.

We are going to select the tables **country** and **statistic** in order to proceed with the SQL query. This selection should display the following screen:

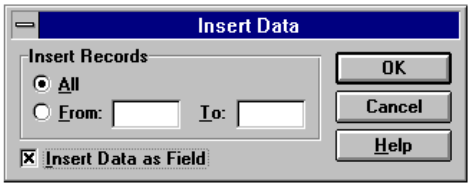


The associated SQL query is as follows:



The data can now be returned to Word in the form of a table by selecting the option **File➔Return data to Microsoft Word** in the main menu and clicking on the button **Insert Data**.

The following window should open: select the check box **Insert As Field**.



The table can now be organized under Word:

Country	1990	1991	1992
Germany	78.03	77.30	79.63
Belgium	9.89	10.01	9.97
Denmark	5.13	5.13	5.14
Spain	39.16	39.32	39.05
France	56.12	56.45	56.68
United Kingdom	57.27	57.48	57.54
Greece	10.04	10.04	10.08
Ireland	3.54	3.50	3.50
Italy	57.54	57.58	57.72
Luxembourg	0.38	0.37	0.38
The Netherlands	14.83	14.93	15.02
Portugal	10.33	10.37	10.39

Using Word mail merge

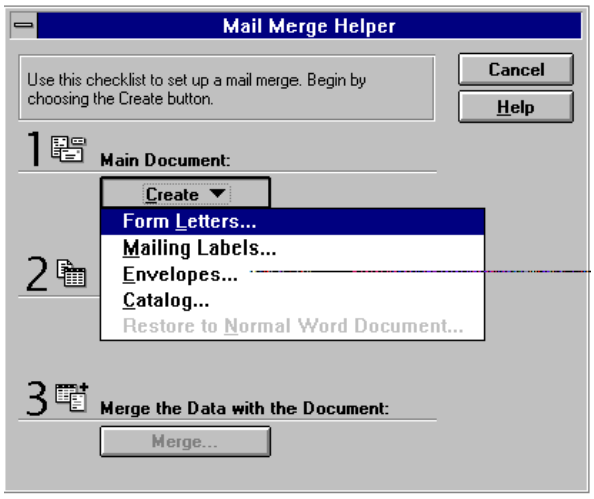
It is very useful to be able to merge data for form letters with different addresses (mailshots) using Word or for standard documents which require different data depending on the particular case.

This is achieved by declaring variables in the text which will be updated when the data is merged.

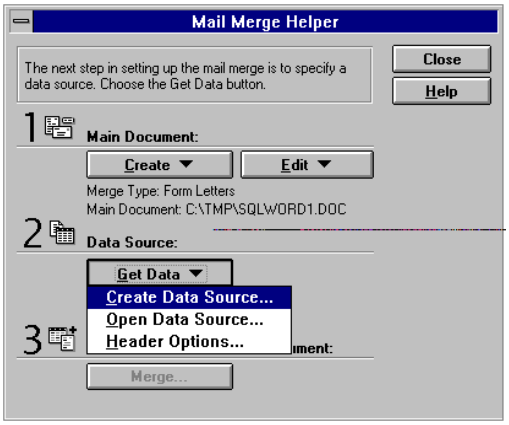
For the example in question, we are going to create a document whose contents are as follows:

- The population is "variable" millions
- The life expectancy is "variable" years
- The birth rate is "variable" %

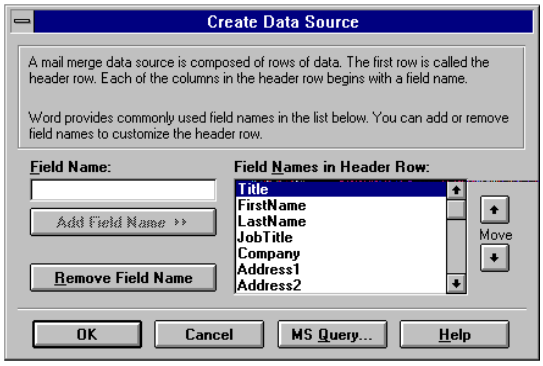
The document will be updated for the year 1993 and for each European Community country. To achieve this, open a new Word document and select the option **Tools➔Mail Merge** and then click on the button **Create** and select the option **Form Letters**.



Select the option **Active Window**. Click on the button **Get Data** and select the option **Create Data Source...**:

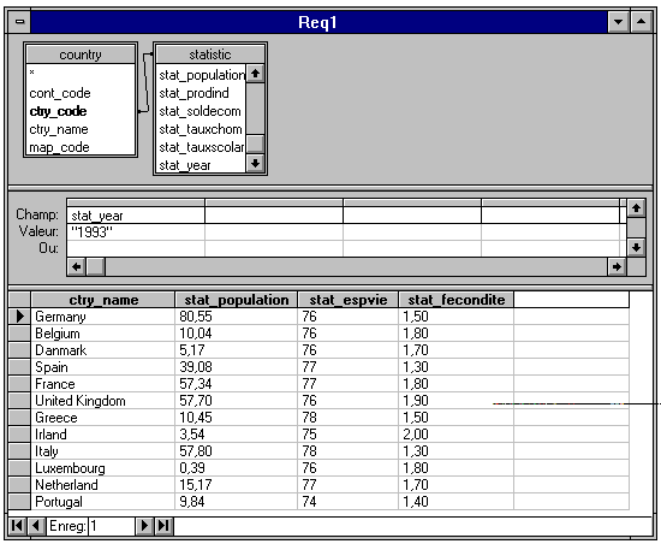


The following window opens:



Since the data source is located on our host, we are going to use MS Query again to transfer the relevant information. Click on the button MS Query.

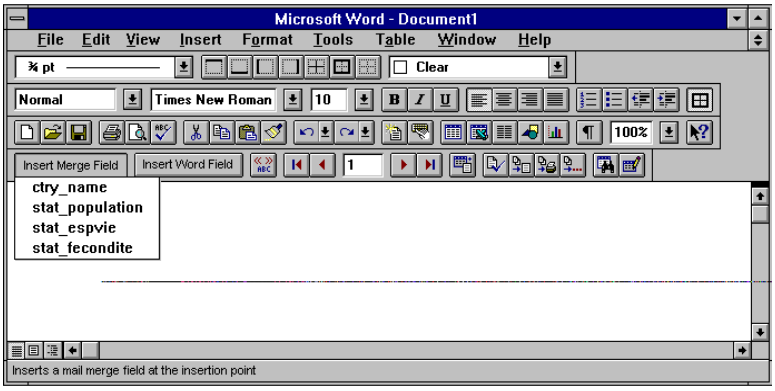
The connection with the host will be established, as for the preceding examples, by selecting the data source and then the relevant tables which include the fields ctry_name, stat_population, stat_espvie and stat_fecondite. The query is summarized in the following MS Query window.



The data may then be returned to Word. When the data is sent back, Word needs to be told to **Edit The Main Document**

In this way, we can edit a new document by inserting "variables" in the text which will be updated from the database by performing a merge operation.

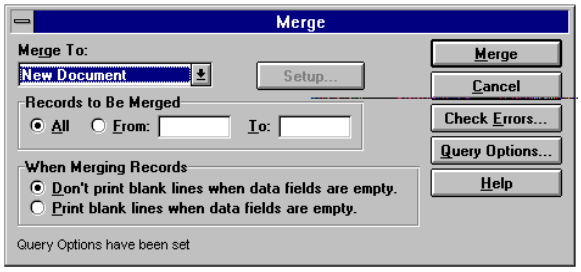
The different fields will be added to the text using the button **Insert Merge Field** which now appears in the main Word screen showing the values of the fields which had previously been selected through MS Query (see figure below).



This allows us, then, to edit the following text by inserting the different variables.

In **crty_name**: - The population is **stat_population** millions
- The life expectancy is **stat_vie** years
- The birth rate is **stat_fecondite** %

These variables are updated for the year 1993 by selecting the option **Tools→Mail Merge** in the general menu, and then the option **Merge**. The following window should appear:



On selecting the option Merge, the form letter is created with the set of different variables in the database.

In our case, this produces:

In Germany: - The population is 80.55 million
- The life expectancy is 76 years
- The birth rate is 1.5%

----Page break---

In Belgium: - The population is 10.04 million
- The life expectancy is 76 years
- The birth rate is 1.8%

----Page break---

In France: - The population is 57.34 million
- The life expectancy is 77 years
- The birth rate is 1.8%

----Page break---

etc.....

USING Tun SQL WITH MICROSOFT VISUAL BASIC

Working principles

Tun SQL, supporting ODBC level 1, can be used to query the database in order to ascertain its structure. The Visual Basic example returns the list of existing tables as well as the list of defined fields for each table for a given database. It can also be used to ascertain the list of values for a field in a table.

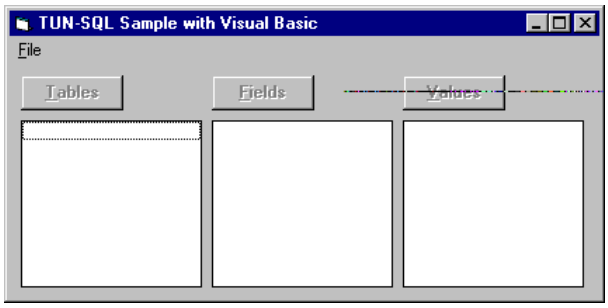
Note:	This example works with Visual Basic Version 3.0, Professional Edition.

Using the example

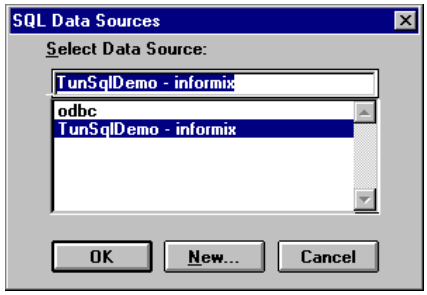
The utility supplied can be activated in two ways:

- From Visual Basic, by loading the file TUNVBSQL.FRM and using the menu option **Run➔Start**.
- Directly by selecting the icon TUNVBSQL.EXE.

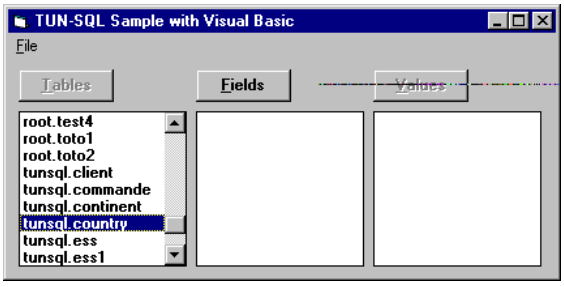
When the example is run, the following window is displayed:



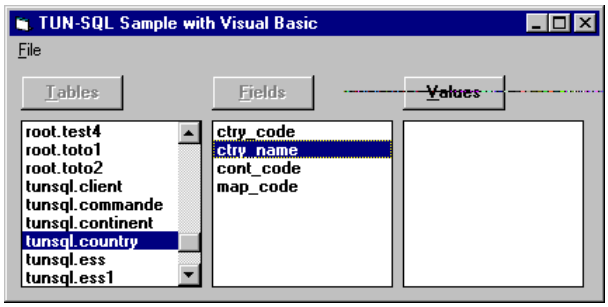
The first operation to perform is to connect to a data source by selecting the menu option **File-➔Open Data Source**. The following window appears:



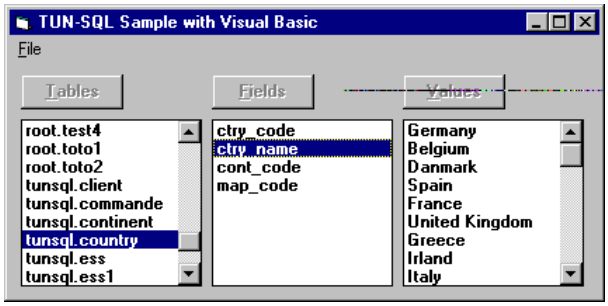
Select a data source and then enter, if necessary, the name of the user and his password so the program can establish the connection. As soon as the connection has been established, the button **Tables...** becomes active. Click on the button to display the list of all the available tables in the database:



If a table is selected, the button **Fields** becomes active; click on this button to display the list of all the fields used in the table.



Similarly, by pressing the button **Values**, the list of possible values for the selected field and table is displayed:



It is then possible to change tables and/or fields to obtain other values.
Note: the display of the "Values" list corresponds to executing the SQL command "SELECT field_name FROM table_name".

Technical considerations

Access to an ODBC driver from Visual Basic is effected using the pre-defined object **Database**.

The function OpenDatabase initializes a database as an ODBC type. Calling this function opens a connection to a data source and returns a Database type of object.

It is then possible to perform a variety of operations on this object such as:

- Ascertaining the number of tables in the database (Database.TableDefs.Count).
- Ascertaining the names of the tables in the database (Database.TableDefs(i).Name).
- Ascertaining the number of fields in a particular table (Database.TableDef(i).Fields.Count).
- Ascertaining the names of the fields in a particular table (Database.TableDef(i).Fields(j).Name).
- Submitting and obtaining the results of SQL queries (Database.CreateSnapShot(SQL query)).

Of course, there are other properties and methods associated with the object **Database**. For further information, refer to the documentation supplied with Visual Basic and, in particular, the more extensive example "visdata.mak".

CHAPTER 6 - REFERENCE GUIDE

INDEX

Note: xxx stands for the relative file extension for a particular database. The file extensions are as follows:

ifx	Informix
ora	Oracle
syb	Sybase
db2	DB2
pro	Progress

CONFIG.XXX	File containing the working and security parameters of the **Tun SQL** UNIX server
DBMAP	Windows application to create or edit character conversion tables.
DBSCRIPT	Windows application which interprets and executes SQL batch files
DBSHOW	Windows application for testing and configuration
PARAM.XXX	File containing the setup parameters of the **Tun SQL** UNIX server
TUNODBC200.XXX	**Tun SQL** UNIX server

CONFIG.XXX CONFIG.XXX

File containing the working and security parameters of the **Tun SQL** UNIX server

Description

The **config.xxx** files can supply a certain number of parameters to the **Tun SQL** UNIX server. Unlike the **param.xxx** files, the parameters do not concern the overall functioning of the server but are refer to a particular database. For example, a **config** file looks like this:

```
#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=DatabaseName
#Version=DatabaseVersion

# 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]
```

This file may contain as many sections as there are databases managed by the DBMS associated with the **Tun SQL** UNIX server. It is not necessary to include all the databases if you have no special parameters to set.

Each section has the name of the corresponding database enclosed in square brackets as a title (for example, **[tunsqldemo]**). The following parameters may be defined in each section:

Define=END_VARIABLE:value

Assigns a value to an environment variable **ENV_VARIABLE** before the opening of a database (installation directory of the database, date format...). This option may be used as many times as necessary in the **config** file.

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

Some business applications leave the user free to compose his own SQL requests. In other cases, applications tend to read a table in its entirety before displaying it on the screen. This does not pose any real problem if it is a question of small tables contained in a local database. However, insurmountable problems may be posed when it is a question of very large tables in a centralized, remote database. In this case, there is a considerable number of exchanges on the network and there is insufficient memory in the PC to store the received data. The PC has to be frequently rebooted after such a **select** request. To compensate for this problem, the **Tun SQL** ODBC driver incorporates the notion of limits which can be defined through the parameter **RowLimitMode**. If this parameter is defined it takes priority over the values which may have been defined in the data source on the PC.

The parameter may have five different values:

None	No limit is imposed by the ODBC driver
Absolute	The ODBC driver will refuse to load more than **RowLimitValue** rows during one **select** request. No messages will be displayed to inform the user.
Fixed	The ODBC driver will refuse to load more than **RowLimitValue** rows during one **select** request. A message will be displayed informing the user.
Variable	The ODBC driver will refuse to load more than **RowLimitValue** rows during one **select** request. It will however display a message proposing to load more within the limit of the maximum value (**RowLimitMax**).
Extended Limit	The ODBC driver will refuse to load more than **RowLimitValue** rows during one **select** request. It will, however, display a message to load more without a maximum value. In this case the message displayed is really only a warning.

DbmsName=DatabaseName

This parameter is for setting or changing the name of the database to transmit to the ODBC driver.

Version=DatabaseVersion

This parameter is for setting or changing the version number of the database to transmit to the ODBC driver.

In addition to the sections corresponding to each database, the **config** file may include the sections **[Allowed]** and **[Denied]** which can be used to secure access to certain databases. These sections function as follows:

[Allowed]

This section must contain a series of triple terms such as:

```
base_name,user_name,product_name
```

where:

- **base_name** is the name of a database
- **user_name** is the name of a user of the database
- **product_name** is the name of a Windows application which will use the server.

Each set of triplets indicates if a user (**user_name**) is authorized to use the database (**base_name**) with the Windows application (**product_name**). Each parameter can be replaced by the generic character *****.

For example, the triplet **tunsqldemp,*, excel** means that all the users can use the base **tunsqldemo** from the application EXCEL except for those whose names are contained in a similar triplet in the section **[Denied]**.

[Denied]

This section should contain a series of triplets of the same type as found in the section **[Allowed]**.

Each set of triplets indicates if a user (**user_name**) is not authorized to use the database (**base_name**) with a Windows application (**product_name**). Each parameter can be replaced by the generic character *****.

For example, the triplet ***,john,excel** means that the user **john** cannot use any database from the application **excel** except for those whose names are contained in a similar, contradictory triplet in the section **[Allowed]**.

Notes: For a **Tun SQL** UNIX server to take account of a **config** file, it must be indicated to the server using the command line option **-c**.

Two different Informix database engines may coexist (Informix version 5 and Informix version 7). For example, one database could be accessed by database engine 1 from the directory /u/informix1 and a second by database engine 2 from the directory /u/informix2. In this case, the file config.ifx contains:

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

See also

param.xxx, tunodbc200.xxx

DBMAP DBMAP

Windows application for creating or editing conversion tables

Syntax

```
DBMAP [-ffile_name]
```

Description

Tun DB Map can be used for creating or editing character translation tables. The translation tables allow the to avoid problems caused by the differences in the codification of accented characters which might occur between the PC and the remote DBMS. For them to be taken into account, the translation tables must be declared in the definition of the relevant data source.

-ffile_name
Is used to indicate the name of an existing translation table.

DBSCRIPT DBSCRIPT

Windows application for executing SQL batch files

Syntax

```
DBSCRIPT [-ddata_source] [file_name]
```

Description

Tun DB Script makes it possible to execute a whole list of SQL requests in a single operation. It interprets the SQL commands one after the other and stops when it encounters an error. **Tun DB Script** can also be used to change and save SQL batch files.

Tun DB Script is useful for downloading the contents of a database from a PC running under Windows to a remote DBMS. It can also be used for the large-scale updating or clearing of databases.

file_name
Is used to indicate the name of the file containing the SQL commands which will be loaded when the application starts running.

-ddata_source
Is used to indicate the name of the data source that the SQL batch file will work on. **Tun DB Script** does not automatically create the connection with the data source. This must be done subsequently "by hand".

DBSHOW DBSHOW

Windows application for testing and configuration.

Syntax

```
DBSHOW [-h host_name]
```

Description

This application can be used to query a UNIX host on the network to see if there are one or more **Tun SQL** UNIX servers present.

Enter the name of the host in the field **Host,** then click on the button **Query Host** to obtain this information. If one or more **Tun SQL** UNIX servers are correctly installed on the remote machine, **Tun DB Show** returns their names and the names of the DBMSs with which they are interfaced.

This application is particularly useful for checking the conformity of the installation.

-hhost_name
Indicates the name of the host to be queried.

PARAM.XXX PARAM.XXX

File containing the setup parameters of the **Tun SQL** UNIX server.

Description

Rather than running **Tun SQL** UNIX servers with a large number of command line options, it is preferable to write all the options one after the other to a file and to transmit the name of the file to the host after the option **-f**.

The **Tun SQL** installation procedure uses this mechanism and writes the options to **config.xxx** files where the character **xxx** is replaced by the abbreviated name of the DBMS (**config.ora**, **config.syb**, **config.ifx**). Here is an example of such a file:

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

The meaning of the different options is explained in the section **tunodbc200.xxx**.

See also

config.xxx, tunodbc200.xxx

TUNODBC200.XXX

TUNODBC200.XXX

Tun SQL UNIX server.

> **Note**: The different **Tun SQL** UNIX servers have a certain number of options in common. The list of the different options can be obtained by calling the executable tunodbc200.xxx with the option **-a[ll]**, where xxx represents the relative extension for the database in question.

Syntax

```
tunodbc200.XXX
   -a[ll]
   -c[config]=config_file
   -Dname=value
   -db[ms]=DBMS_name
   -de[bug]
   -f[ile]=param_file
   -h[old]
   -i[nter]
   -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...
```

Description

-a

Lists all the options supported by **tunodbc200.xxx.**

-c=config_file

Associates a configuration file (**config.xxx**) with the server (Cf. config.xxx).

-db=name

Is used to associate a DBMS name with the **Tun SQL** UNIX server. This is the value which will be displayed in the list when **Tun DB Show** is run.

-de

Tells the server to function in trace mode. The messages will be displayed by default on the device **/dev/console**.

-Dname=value

Sets the environment variable "**name**" to the value "**value**" before the execution of the server (installation directory of the database, format of the date...). This option may be repeated as many times as necessary in the command line. It is absolutely indispensable to define some of the variables so that the **Tun SQL** UNIX server can function with certain DBMSs. This definition is carried out by the installation procedure. For reference purposes, these variables are needed for the following DBMSs:

Oracle:

ORACLE_HOME: Oracle installation directory.
ORACLE_SID: Default database.

Informix

INFORMIX_DIR: Informix installation directory.
DBPATH: Directory containing the database (only SE).

Sybase

SYBASE: Sybase installation directory.
SYBSERVNAME: Identifier of the server setup file (optional). This value is equivalent to the variable DSQUERY which is defined and used by SYBASE.

-f=param_file
Indicates the name of a file in which all the options defined for the program are written consecutively. This option can be used to run the program without having to enter dozens of different options.

-i
Uses interactive test mode to check the server is working properly.

-o=file
Indicates the name of the file or the device to which the server trace messages and its access controller (Watchdog) will be written. Works only in **debug** mode.

-o2=file
Indicates the name of the file or the device to which only the Watchdog trace messages, and not those of the server, will be written. Works only in **debug** mode.

-t=xx
Assigns a timeout value. This is the time after which any lack of reply from the PC will be considered as a PC stoppage. The **Tun SQL** UNIX server will therefore also cease to function. This value is of lesser priority than that which is assigned on the PC when the data source is defined.

-u
Used with the following parameter to define the list of authorized users ("*****" for everyone). The default value is *****. For example:

```
-x = *
-u = bill          (only "bill" is authorized)
```

-v=XX
Associates a DBMS version number with the **Tun SQL** UNIX server. It is a value which will be displayed in the list when **Tun DB Show** is run.

-x
Used to define the list of users denied access ("*****" for everyone). For example:

```
-u = *
-x = bill          (only "bill" is denied access)
```

Informix options

-h
By default, Informix does not maintain the cursors open during the execution of the commands **commit** and **rollback**. The option -h will ensure the cursors are maintained open after such a command if the applications using this type of server are unable to do so.

-n
Applies only to SCO UNIX 5. If the system SCO UNIX 3.2 version 5 cannot correctly check user passwords, this option cancels password checking.

-s
By default, Informix cannot carry out **select** commands with a sort option (**group by** or **order by**) on a column which has not been included in the **select** command. The option **-s** can be used to compensate for this lack when applications make no provision for this.

Oracle options

-l

If tables, columns, indexes or views have been created in the catalog with lower case characters, the option -l ensures that the catalog functions return data enclosed in quotation marks. Applications using this server will create queries with names in quotation marks.

Progress options

-n

Applies only to SCO UNIX 5. If SCO UNIX 3.2 version 5 cannot check user passwords correctly, this option cancels password checking.

-nor

The Progress database cannot tell how many lines have been modified or deleted when an **update** or **delete** command is executed. By default, this server compensates for this lack. However, there is a time penalty each time an **update** or **delete** command is given. If applications using the server do not need to know the number of modified lines, the option **-nor** cancels server compensation, thus saving time.

-ow

By default, the server does not support the notion of object owner in the database. In fact, some applications try to add an owner prefix when owners are returned with the objects, thus provoking errors during execution. This option, add owners, is useful when the application needs to know the owner and the owners are correctly handled.

-p=XX

If options that are specific to Progress have to be used (for example, the option -Q which forces the database to respect the ANSI norm), -p=XX should be used, with **XX** replaced by a string containing the required options in quotation marks.

-sv

The Progress database uses only one type of character string. By default, all the strings are considered to be of a fixed length (SQL_CHAR in ODBC). If this option is used, the strings will be treated as though they were of variable length (**SQL_VARCHAR** in ODBC).

-sy

Progress can define system or hidden columns which are not returned when a search of all the columns are carried out using a wildcard character. This is why the server does not describe the columns in its catalog by default. If, however, the hidden columns are required, this option enforces their return.

See also

param.xxx, config.xxx

PART 3
TUN DB REVAMP

CHAPTER 7 - DATABASE REVAMPING

VIRTUAL DATABASES

Most of today's structured data storage consists of Relational Database Management Systems RDBMS. Databases make it possible to store the corporation's data resources which can then be updated using special applications. The mass of data assembled in this way is also of interest to a large number of users wishing to extract from these data reservoirs the information they require for their occupation (performance indicators, statistics, expert systems). The SQL language is used for updating and querying databases.

However, the structure of databases, which are at the heart of the information system, can complicate access to the information they contain at every level of the corporation:

- There is a considerable number of tables and records in databases, too many in fact for the average end-user who is often only interested in a part of the data.
- Database structure is always complex and the user requires a lot of experience to find his way around.
- The computing environment of databases is not very user-friendly. For example, the names of the tables and records are seldom expressed in straightforward terms.
- Data manipulation and use require prior knowledge of the SQL language to query databases and obtain the desired results.

Several breakthroughs have been made to reduce these obstacles and facilitate access to databases (e.g. the inclusion of graphical interfaces in database interrogation tools).

The next step is aimed at freeing the end-user totally from the pre-requisite of technical understanding of databases by only making available the information he needs in the most suitable form for his work environment.

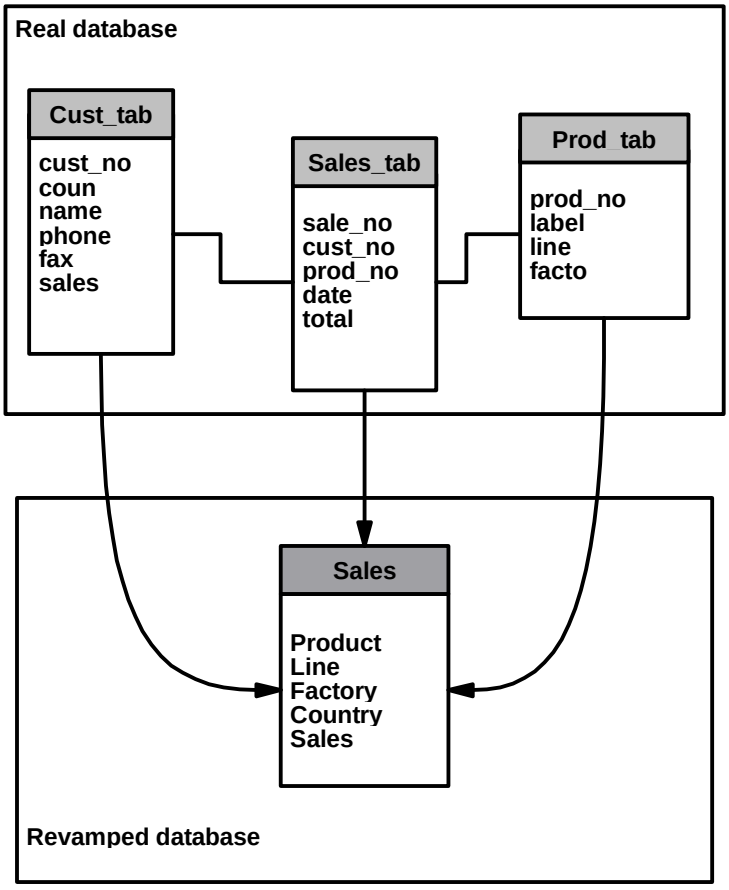
The consequences of such a change are:

- Improved productivity: the end-user becomes autonomous in his use of data, analysis and decision-making take less time because they are easier.
- More relevant information: with only the data he needs and can master, the user sharpens his capacity for analysis and synthesis and refines his results.

Revamping

The principle of revamping consists of constructing a virtual database adapted to the user's environment from the existing database. Although it does not exist as a real database, the new structure is viewed by the user as a normal database whose tables and fields, however, correspond exactly to his needs: the database only contains the information the user really needs for his analyses, in a form which suits his requirements (understandable data names, predefined functions).

The redefined database is perfected by an administrator who reconfigures the tables and fields from real databases. For example:



In the above example, the real database contains three tables: "Cust_tab" (table of customers), "Sales_tab" (table of sales) and "Prod_tab" (table of products).

The administrator defines a virtual table presenting the results of sales per product, per product line, per production site and per country.

The virtual table entitled "Sales" contains the following fields:

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

The virtual table constructs a join between the tables "Prod_tab" and "Sales_tab" using the common field "prod_no", and a join between the tables "Sales_tab" and "Cust_tab" using the common field "cust_no".

REVAMPING IN Tun SQL

Tun SQL can perform the administration and use of virtual databases because of the following two components:

- The administrator of databases **Tun DB Revamp** which manages the revamping.
- The virtual ODBC driver which allows the user to access databases revamped by the administrator.

The DB Revamp administrator

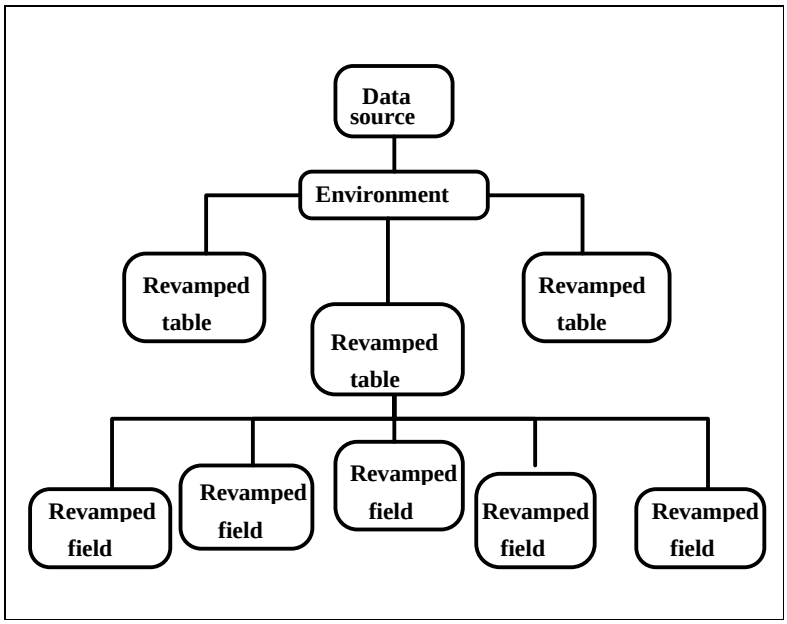
The objective of the **Tun SQL** virtual database is to offer the end-user contextually redefined information for a particular "environment".

Thanks to an intuitive graphic interface, the administrator can define as many "environments" as he wants for different users or types of user. The environment is based on "occupation": an accountant might only see the tables related to accountancy, sales staff only the tables related to their main activity.

Each environment can be viewed by the ODBC front-end which makes the query as a special data source (refer to the architecture diagram in the section **Virtual ODBC driver**).

The virtual database model is as follows:

- One or more environments defined from a real data source which contain a selection of particular tables in the real database based on the end-user's needs.
- The tables defined in an environment are either native tables from the real database or the result of joins between two or more tables (notion of view).
- Each table contains the fields required by the end-user and only those.
- The revamped fields are either existing fields from real tables or recalculated fields that further simplify the end use of the database.
- The revamped tables and fields can be renamed with names that more explicit for the end-user (e.g. "Cust_tab" might become "Customer Table" and "Cust_no" "Client Number").



The tables redefined in an environment do not physically exist in the database. However, the revamped database is stored in an indexed form in three supplementary tables created in this database:

- The environment table containing a list of environments in the form "environment name" and "description".

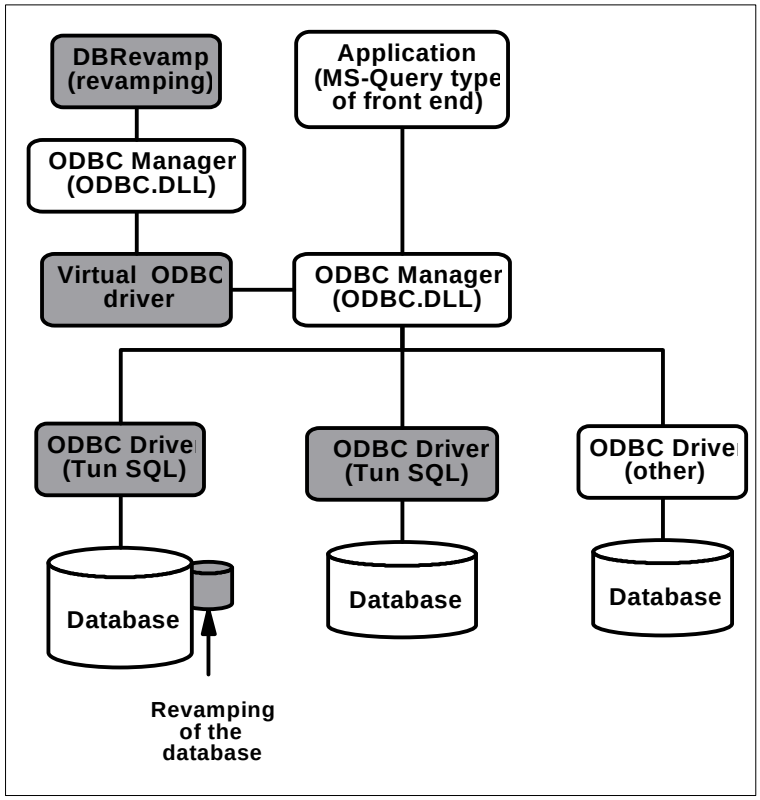
- A table containing the list of redefined tables, each of which is indexed by a name, a description and the name of the environment to which it belongs.
- A table of redefined fields, each of which is indexed by a name, a description, an origin (existing field, processed data, data concatenation) and the name of the virtual table to which it belongs.

After a revamping operation, a database will always contain these three extra tables.

Virtual ODBC Driver

For the end-user, querying a virtual database is similar to querying a real database in read-only mode. This transparency is due to the integration into **Tun SQL** of a special ODBC driver for virtual databases.

When the ODBC manager (ODBC.DLL) receives queries from a particular environment, it transmits them to the virtual ODBC driver whose task is then to translate them into appropriate queries for the real database. Next the virtual ODBC driver passes the translated queries back to the ODBC manager which directs them to the real database's normal ODBC driver.



CHAPTER 8 - PRACTICAL REVAMPING

The present chapter explains how to produce a sample virtual database to give the administrator practical experience of how **Tun DB Revamp** functions.

Each step in the process of revamping a database with **Tun DB Revamp** is explained and, when necessary, illustrated.

The revamping of a database starts with a data source which identifies the ODBC driver used to manage queries to the initial database. This operation, therefore, requires the prior definition of the data source associated with the database that the administrator is to revamp. The example discussed in this chapter starts with a data source that has already been defined in **Tun SQL** and selected in **Tun DB Revamp**

Refer to the sections "**Creating a data source**" in the chapter entitled "**Configuration**" and "**Importing a data source**" in the chapter "**Tun DB Revamp General Use**".

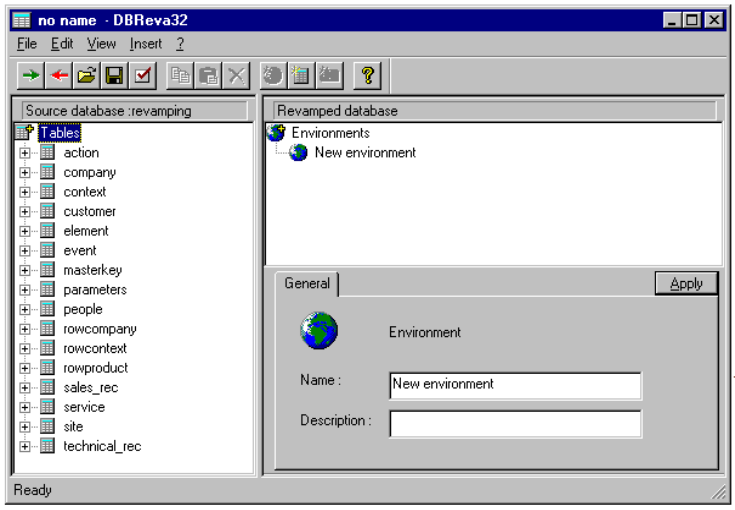
OBJECTIVES OF REVAMPING

Starting with a real database, we shall construct an environment which assembles in two virtual tables the information required by a given group of users.

The real database used for this example contains data relating to customers and products spread over about fifteen tables. The objectives of the revamping are:

- To simplify access to the database by regrouping in the one environment the data the users need and leaving out data that is irrelevant to their needs.
- To rename the data so it is more easily understandable.
- To customize data presentation by presenting information with a higher degree of synthesis than is offered by real tables: this is obtained by creating a virtual field which contains the result of a function applied to real fields.

After importing the data source "revamping" (refer to the chapter entitled "**Tun DB Revamp General Use**" for how to import a data source), the example begins as follows:



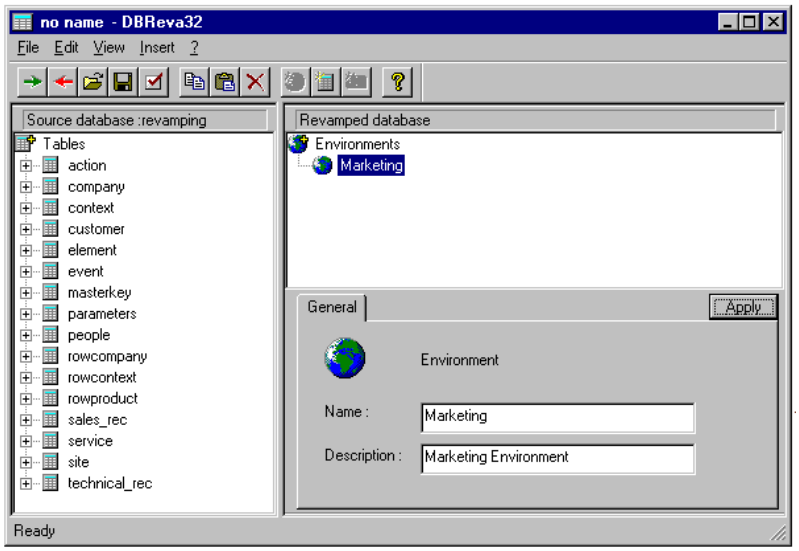
The contents of the database attached to the data source appears in the left-hand section of the window and the initial directory structure of what will be the virtual database in the right-hand section.

DEFINING AN ENVIRONMENT

When a data source that has not been previously defined is opened, a default environment is created by **Tun DB Revamp**. We shall now configure this environment.

If the option **Display➔Property Box** is active (which it is by default), the properties dialog box remains permanently displayed to allow modification of the object properties.

The administrator calls the environment "Marketing" by changing the name corresponding to the current environment on the General tab. He adds the description "Marketing Environment".



The environment is now ready to receive its virtual tables.

COPYING A REAL TABLE

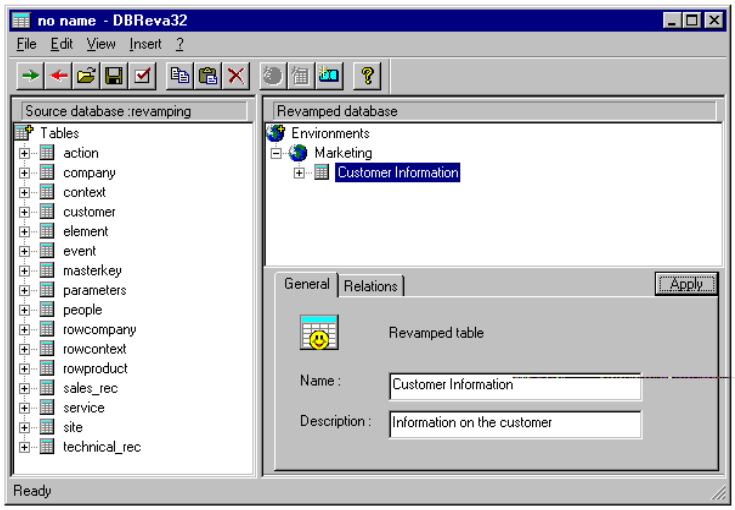
The users of the environment "Marketing" frequently need to access data related to the company's customers. They especially want to obtain customers addresses and function quickly. Since this information is located in the table "Customer" in the real database, the administrator can copy the table into the "Marketing" environment, rename the table, delete particular fields and rename others.

Copying the real table

The simplest way to copy an object is with "drag 'n drop". The other methods are described in the section "General Remarks" in the chapter "**Tun DB Revamp General Use**".

The administrator performs the following operations:

- Select the real table "customer" in the left-hand window and drag 'n drop it to the environment "Marketing" in the right-hand window.
- Change the name of the virtual table "customer" to "Customer Information" on the **General** tab and add the description "Customer Information" (it is quicker to press the F2 function-key to rename the table).
- Click the button **Apply**.

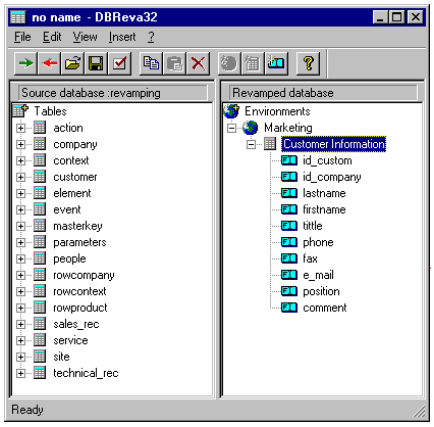


Choosing virtual fields

Some fields in the real table hold no interest for particular users. The administrator should remove these fields from the virtual table. Additionally, the fields which are kept in the virtual table should be renamed to give users a better idea of their contents.

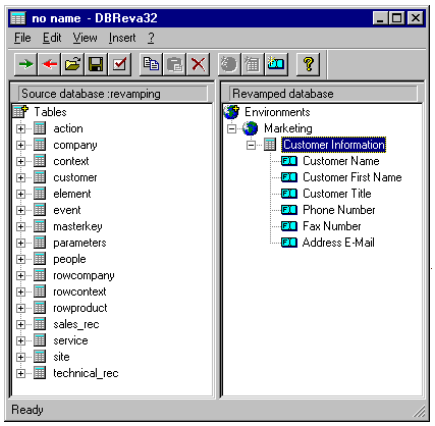
To achieve this, the administrator performs the following steps:

- Click the sign ⊞ in front of the virtual table object to view the fields contained in the virtual table.
- Select and remove the unnecessary fields one by one using one of the methods described in the section "**General Remarks**" in the chapter "**Tun DB Revamp General Use**".
- Rename the retained fields individually using the function-key F2, the "Enter" key and the arrow keys to move from one field to another.



After studying the fields retained in the virtual table, the administrator decides to delete the fields "id_custom", "id_company" and "position" which are identification codes, and also the field "comment" which the users do not need. He changes the names of the following fields from:

- "lastname" to "Customer Name",
- "firstname" to "Customer First Name",
- "title" to "Customer Title"
- "phone" to "Phone Number",
- "fax" to "Fax Number",
- "e_mail" to "E-mail Address".



INSERTING NEW REAL FIELDS

As some information required for the virtual table "Customer Information" is not found in the real table from which it was copied, the administrator has to copy fields from other real tables into the virtual table.

The administrator wants to add the following information to the virtual table:

- the name of the company the customer works for
- the type of business the customer's company is involved in
- a description of his function

The fields containing the customer's company's business and the description of his functions are in a particular table with the descriptions indexed by different codes in different parts of the database. This, and inclusion of the names of customers' companies, will be dealt with in the section "Special Cases" which follows the next.

General additions

To add a new field for companies' names, the administrator has to carry out the following steps:

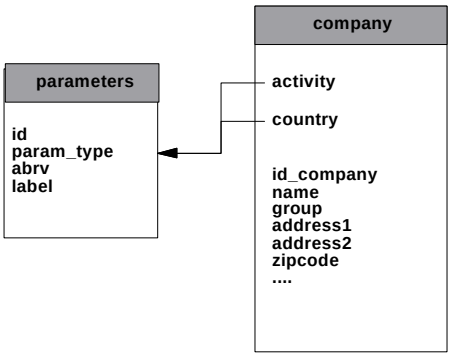
- Select the field "name" in the real table "company" in the left-hand area of the window and drag 'n drop it to the virtual table "Customer Information" in the right-hand area.
- Change the name of the field "name" to "Customer Company" using function-key **F2**.

Special cases

In the present example, there is a special table ("parameters") in the physical tables which contains the uncoded descriptions that are indexed by codes in different parts of the database.

Example:
The table "company" contains the field "activity" which uses a unique numerical code to identify the business activity of the customer's company; it also contains the field "country" with a unique code for each country. The table "parameters" establishes the relationship between these codes and the uncoded descriptions to which they correspond by means of two links with the table "company":
the first links "company.activity" with "parameters.id"
the second links "company.country" with "parameters.id".

The one physical table represents the two logical tables.



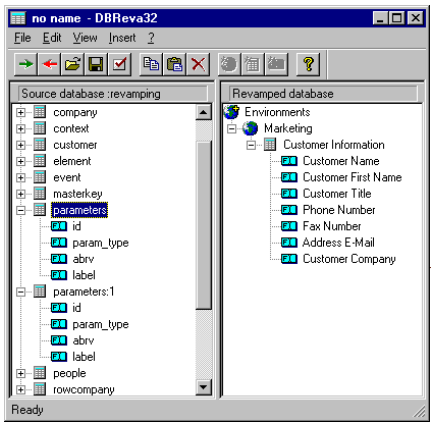
To find the descriptions corresponding to a company's business activity and a customer's function, the administrator refers to the field "id" in the table "parameters" for the fields "activity" in the table "company" and "position" in the table "customer". To do this, he has to work on the logical structure of the database and define two parameter tables with different names. These tables enable him to identify the real field from which the virtual field is derived and also allow the joins between the tables to be made at a later stage (Cf. the section entitled "Definition of inter-table links".

Duplicating the description table

To create two logical tables from a physical "parameters" table, the administrator has to make a copy of the real physical table. To achieve this, he has to perform the following operations:

- Select the real table "parameters" in the left-hand area of the window
- Display the contextual menu related to this object by clicking the right mouse button and choose **Copy** and then **Paste**

The table "parameters:1" is created. It contains exactly the same data contents as the table it was copied from.

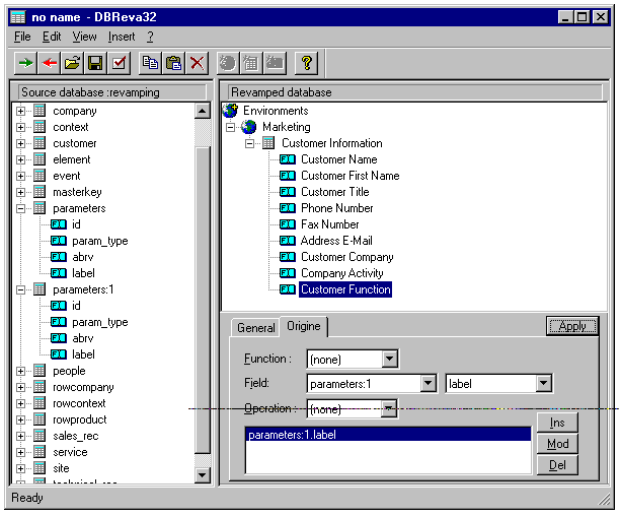


Defining new fields

To add to the virtual table, the fields containing the description of the company's name and the description of the customer's function, the administrator has to follow these steps:

- Create a new field using either the main menu option **Insertion➔New Field** or the option **New Field** in the contextual menu of the virtual table "Customer Information".
- Enter the name "Company Activity" for this field on the **General** tab.
- Click the tab **Origin**.
- Select the real table "parameters" and the field "label" in the **Field** list box.
- Click the button **Mod** to replace the default origin field.

- Click the button **Apply**.
- Repeat these steps for a new field "Customer Function" with "parameters:1.label" as origin.



DEFINING INTER-TABLE LINKS

The addition of fields from different real tables in the virtual table requires the definition of links between these tables so that inter-table joins can be created when the end-user effects a query. In fact, the fields "Customer Company", "Customer Function" and "Company Activity" do not belong to the same table as the other fields in the virtual table. These links may be direct or indirect (link from one table to another, or links from more than one table to other tables).

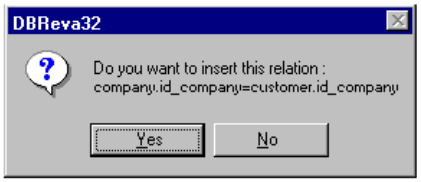
Two complementary methods may be used to define inter-table links.

Defining links automatically

The first method consists of clicking the button  on the tab **Relations** in the virtual table.

Tun DB Revamp will then detect the missing, currently undefined, inter-table links. When a link is missing between two tables, **Tun DB Revamp** tries to link the tables using two fields of the same name. If the two fields exist, **Tun DB Revamp** proposes to use them to link the tables with them.

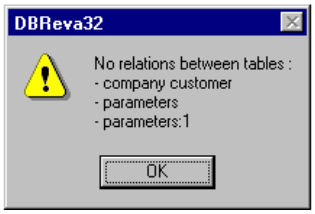
In the present example, where no link has yet been defined, clicking on the "magic wand" button displays the following window:



This means that **Tun DB Revamp** has detected that the real tables "company" and "customer" were used to define the virtual table without being linked directly or indirectly. The program proposes to define a link between fields in the two tables with the same name "id_company".

As this choice of link corresponds to the actual situation, the administrator answers "Yes".

Next, the following dialog box appears:



This means that **Tun DB Revamp** has detected an absence of link between the tables "company" and "customer", and the tables "parameters" and "parameters:1", and that there are no fields in these tables that would allow such a link.

In this case, the administrator has to use the second method.

Defining links manually

The second method consists of defining the links between the two real tables individually to create a virtual table. So as not to leave any out, the administrator has to look closely at the real tables that contain the fields of the future virtual table and for which there are no links with the other tables. These tables are those returned by **Tun DB Revamp** in the preceding dialog box.

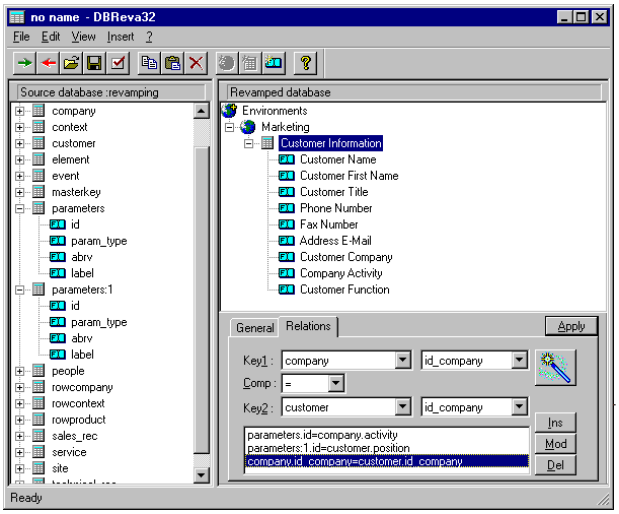
As far as the field "Customer Company" is concerned, the table "company" from which it comes is already linked to the table "customer" from which the other fields in the virtual table, apart from "Customer Company" and "Customer Function", come.

The field "Company Activity" comes from the table "parameters". The administrator has to create a link between the field "id" in this table and the field "activity" in the table "company".

The field "Customer Function" comes from the table "parameters:1". The administrator has to create a link between the field "id" in this field and the field "position" in the table "customer".

To achieve this, the administrator performs the following operations:

- Display the tab **Relations** for the virtual table.
- Select the table "parameters" and the field "id" in the list box **Key1**.
- Select the comparison operator "=".
- Select the table "company" and the field "activity" in the list box **Key2**.
- Click the button **Ins** to insert this link in the list.
- Select the table "parameters:1" and the field "id" in the list box **Key1**.
- Select the comparison operator "=".
- Select the table "customer" and the field "position" in the list box **Key2**.
- Click the button **Ins** to insert this link in the list.
- Click the button **Apply** for the additions to be taken into account.



To check that the real tables are correctly linked, the administrator clicks the button  again.

If the tables are correctly linked, the following window is displayed:

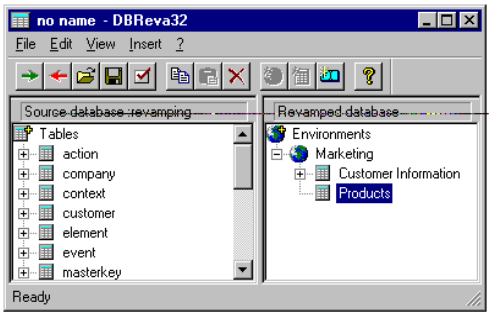
DEFINING A NEW VIRTUAL TABLE

The users of the environment "Marketing" also wish to be able to access information on product sales, especially the number of products sold per sector and the number of PCs installed per product (the current example concerns PC software).

Since this data is spread around several tables, or else is not present in the database in the particular form required (e.g. the number of PCs installed per product), the administrator decides to define a new virtual table and its relevant fields.

To define a new virtual table, the administrator has to carry out the following operations:

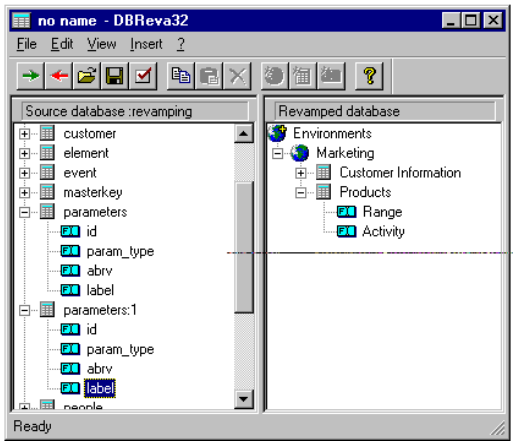
- Create a new table in the environment "Marketing" using either the main menu option **Insertion➔New Table** or the option **New Table** in the contextual menu for the environment.
- Enter the name "Products" for the table on the tab**General**.
- Click the button **Apply**.



Next, to fill the table, the administrator copies the real fields and inserts them directly into the virtual table and then creates the new fields which do not exist in the real database.

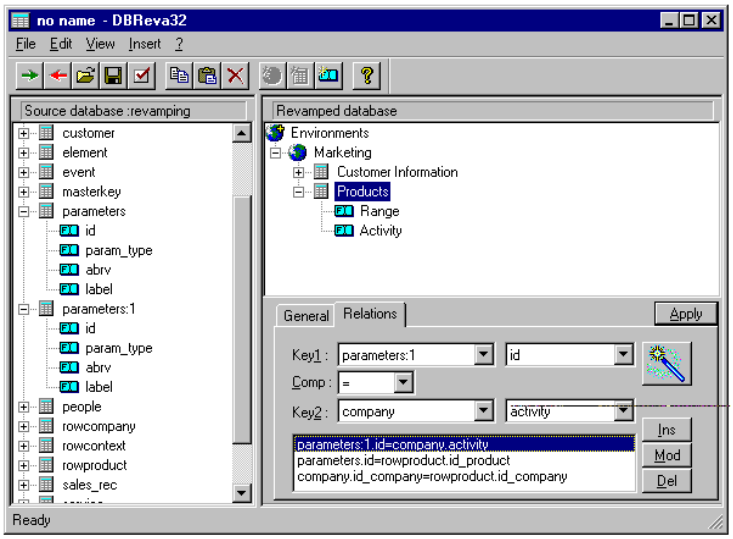
The real fields can be copied as for the table "Customer Information" by drag 'n drop; they must then be renamed. In the present example, the administrator copies:

- the field "label" from the table "parameters" which he renames "Range"
- the field "label" from the table "parameters:1" which he renames "Activity"



Then he creates the following inter-table links:

- "parameters:1.id=company.activity"
- "parameters.id=rowproduct.id_product"
- "company.id_company=rowproduct.id_company"



Refer to the following section for information on defining the calculated field.

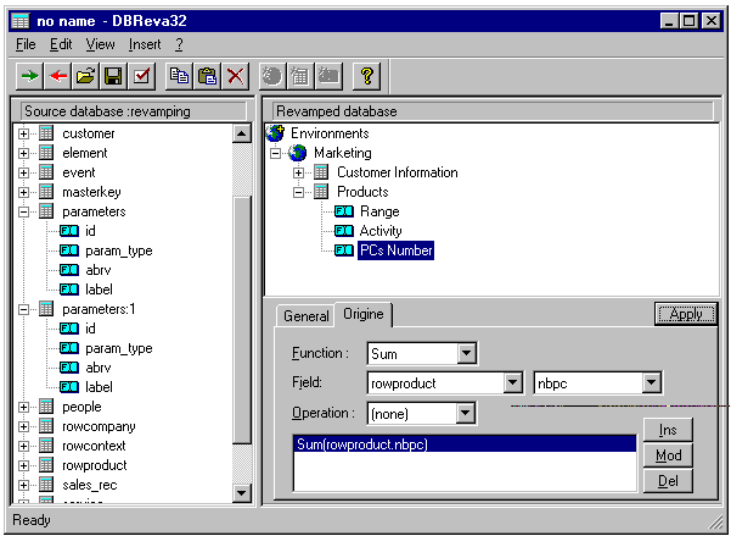
DEFINING A RESULT FIELD

To enable users to access directly the number of PCs equipped with a given product, the administrator has to define a new field in the virtual table "Products" which will contain the result of the function **Sum** after it is applied to values in a real field.

In the real database, there is the field "nbpc" in the table "rowproduct" which indicates the number of PCs equipped with a given software product in a given company.

To define a virtual field containing the number of PCs installed with a particular product, the administrator has to carry out the following steps:

- Create a new field in the virtual table "Products" in the way previously described (using the main menu option **Insertion➔New Field** or the option New Field in the contextual menu of the virtual table), and call it "Number PCs" on the **General** tab.
- Display the tab **Origin** for the field.
- Select the function **Sum** in the **Function** list box.
- Select the table "rowproduct" and the field "nbpc" in the **Field** list boxes.
- Click the button **Mod** to replace the default value with the selection.
- Click the button **Apply**.



Since the real table "rowproduct" is already linked to the other tables used in the virtual table, it is not necessary to add the link.

EXPORTING THE DATA SOURCE

As the environment is now ready, the initial data source can be made available to the users with the main menu option **File➔Export Data Source**. This creates a data source linked to the environment "Marketing" and also three new tables in the database:

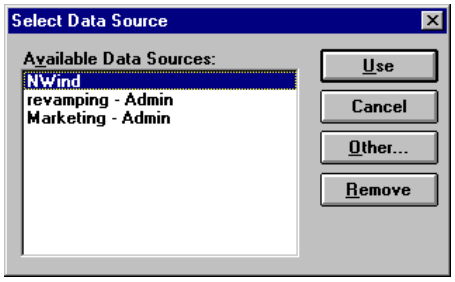
- The environment table: in the present example, it only contains one item.
- The table of tables: in the present example, it contains two items.
- The field table: in the present example, it contains twelve items.

CHAPTER 9 - USING A VIRTUAL DATABASE WITH MS QUERY

This chapter illustrates how to use a virtual database with Microsoft Query. It refers to the example virtual database produced in the previous chapter.

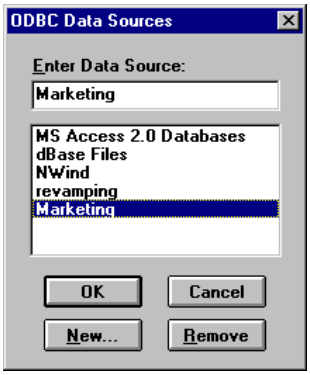
SELECTING THE DATA SOURCE

After starting MS Query, the user makes a new request with the main menu option **File➔New Query**.



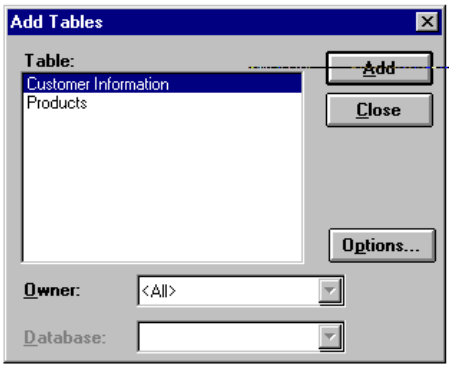
The user selects the data source "Marketing", that is to say the environment defined for him by the administrator with **Tun DB Revamp**

When MS Query is first used to exploit the virtual database, the data source "Marketing" does not appear in the list of available data sources. The user should click the button **Other** to access the following dialog box and select the data source from amongst those listed or add a new one by clicking the button **New**:

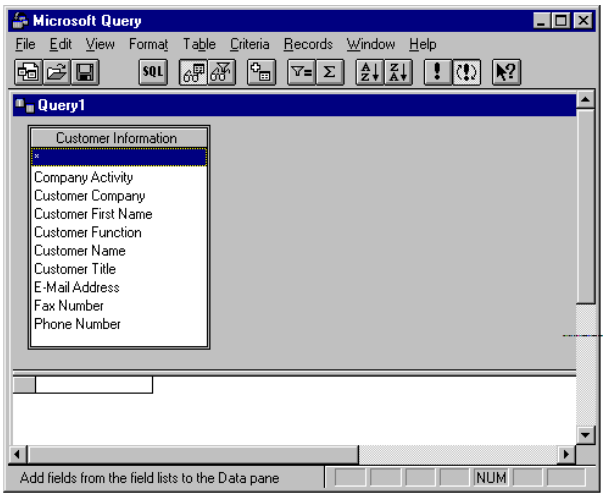


QUERIES WITH MS QUERY

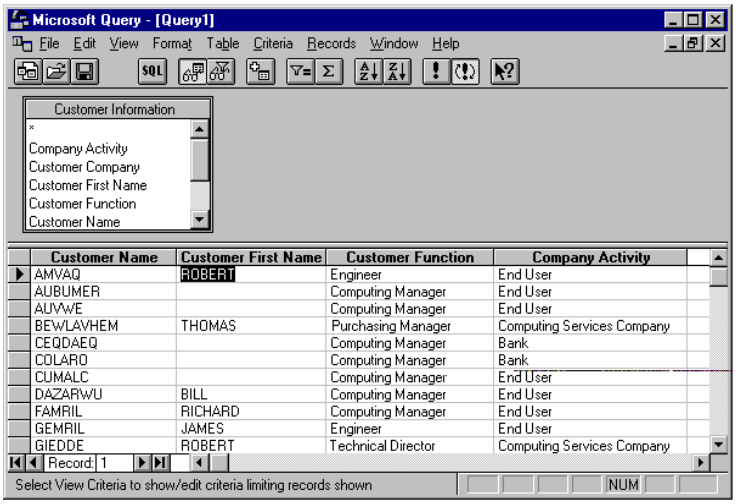
MS Query will then propose a choice of tables from the environment "Marketing".



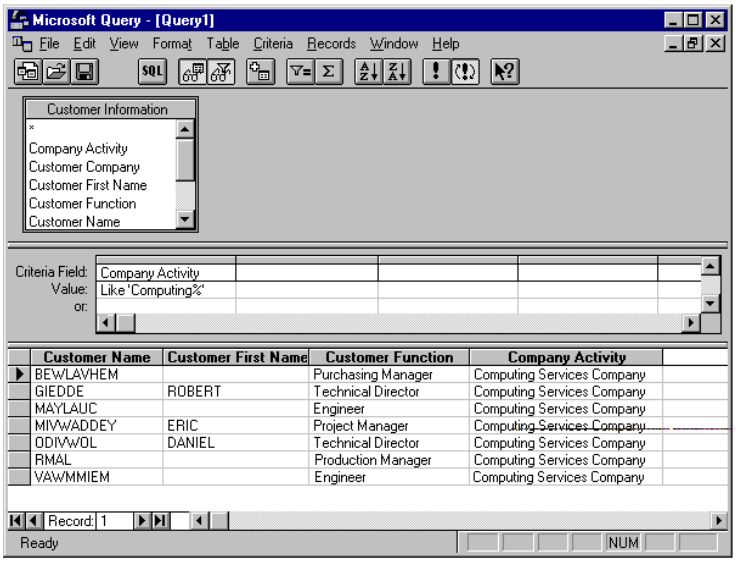
Let us assume the user decides to work on the customer table. He selects the table "Customer Information" and clicks the button **Add**. This displays the following window:



The user can then pass his query. He wants to view the list of his clients (name and surname) as well as their function and the company's activity.
He successively selects the fields "Customer Name", "Customer First Name", "Customer Function" and "Customer Activity". The result of his query is displayed as follows:



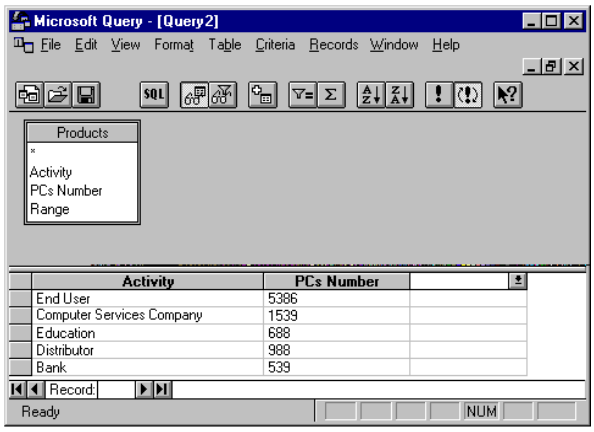
Next he wants to view only the list of customers whose activity is "Computing Services Company". He then defines the search criteria "Company Activity". The result of this query is as follows:



The user next wants to formulate a query for the other table in the environment "Marketing", this time about products. He chooses the table "Products" from the "Marketing" data source.

The user wants to know the number of PCs equipped with his products sorted by the activity of the customers' companies. Since the number of PCs is a pre-calculated field in the virtual table, the user need only select the virtual field "Number of PCs" for the addition (sum) operation to take place automatically with the help of the virtual ODBC driver.

The result of his request is as follows:



CHAPTER 10 - TUN DB REVAMP GENERAL USE

This chapter provides a description of the principal **Tun DB Revamp** commands for general use.

GENERAL OPTIONS

Choosing the working language

To choose the interface language you wish to use, click the option **?➔Language"** and select the language you wish to work in.

Modifying the display

To modify the controls displayed in the main **Tun DB Revamp** window:

- Select or cancel the option **View➔Toolbar** from the main menu to display or hide the toolbar.
- Select or cancel the option **View➔Status Bar** from the main menu to display or hide the status bar.
- Select or cancel the option **View➔Property Box** from the main menu to display or hide the object properties bar.

Copying an object

Use one of the following methods to copy an object:

1. To use the "**drag and drop**" method, select the object to be copied, and drag the mouse to the place you want to copy it to with the mouse button held down.
2. Use the main menu option **Edit➔Copy** to copy the selected object, and then the option **Edit➔Paste** to paste it in the desired location.
3. Use the options **Copy** and **Paste** in the contextual menu (displayed by clicking the right mouse button) to copy the selected and object and paste it in the desired location.
4. Use the keyboard keys **Ctrl-C** (copy) and **Ctrl-V** (paste) to perform the operation.
5. Use the toolbar buttons,  (copy)  (paste), to perform the operation.

Deleting an object

To delete an object, select it by clicking it and then do one of the following:

1. Use the main menu option **Edit➔Delete**.
2. Use the contextual menu option **Delete**.
3. Use one of the keyboard **Del** keys.
4. Click the button  in the toolbar.

Renaming an object

To rename an object, first select it and then use one of the following methods:

1. Use the **General** tab in the property box.
2. Use the **F2** function-key on the keyboard and replace the old name with the new one.
3. Click the object again and proceed as for method 2.

Saving changes

To save changes made to property values, press Enter with the cursor situated in the relevant dialog box, or use the button **Apply**.

Obtaining help

To access the on-line help or obtain more information about **Tun DB Revamp**, click the main menu option **?➔About DBRevamp** or use the toolbar button .

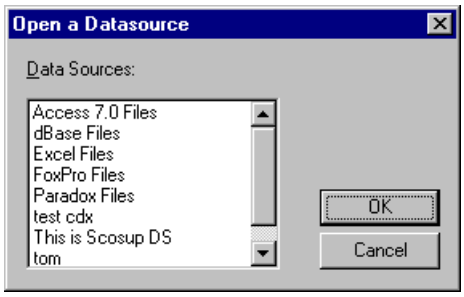
Quitting Tun DB Revamp

To quit the application, click the option**File➔Exit**.

IMPORTING A DATA SOURCE

To redefine (revamp) a real database, you have to select a corresponding data source. You do this by using the option **File➔Import Data Source** or else by clicking the button in the toolbar.

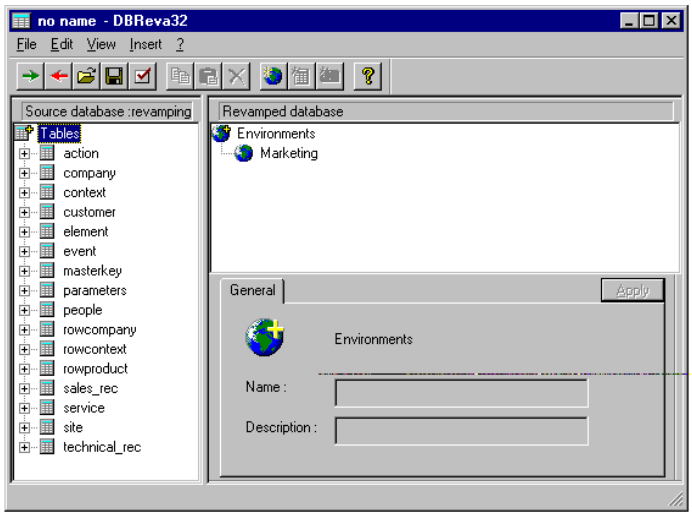
The following dialog box is displayed:



It contains a list of data sources corresponding to the real databases existing on the **Tun SQL** servers. Since the virtual data sources, which are obtained by revamping a real database, cannot be redefined, they do not appear in this list. They do, however, appear in the list of data sources made available to the end-user in any Windows application that uses them (e.g. Microsoft Query).

Select the data source you wish to use.

A **Tun DB Revamp** window similar to the following appears:



The tables of the real database appear in the left-hand section of the window.

If the database in question has not been revamped with **Tun DB Revamp**, the right-hand section of the window will contain an empty environment named "New Environment" which is the first environment you can configure.

On the other hand, if the database has already been revamped with **Tun DB Revamp**, in which case it is a question of the updating the virtual database), the right-hand section will contain the list of environments already created and their contents.

CREATING AN ENVIRONMENT

To define a new environment for the selected data source, click the option **Insert➔New Environment** or the toolbar button  .

Enter a name and, optionally, a description for this environment.

DEFINING A VIRTUAL TABLE

To define a virtual table in an environment, click **Insert➔New Table** or choose the option **New Table** in the contextual menu, or use the toolbar button  .

On the **General** tab of the Properties Bar, enter a name and, optionally, a description for the table, or use the function-key F2 to rename the table.

If you want the virtual table to contain all or part of a real table, you can copy the real table into the environment of your choice; all the fields in the real table will be also copied. To achieve this:

- Use one of the methods described in the introduction (**drag 'n drop**, **Copy/Paste**, keyboard shortcut and toolbar button) to select the real table in the source database and copy it to the target environment.
- Delete the fields you do not require in the virtual database or change them as described in the section "**Defining a field**".
- If you want to, you can change the names of the objects copied (tables and fields) and attach a description to them on the corresponding**General** tab.

DEFINING A FIELD

In a virtual table, you can:

- Insert a field which already exists in a table in the real database, without changing its definition.
- Define a new field from the fields in the real database.

Existing fields

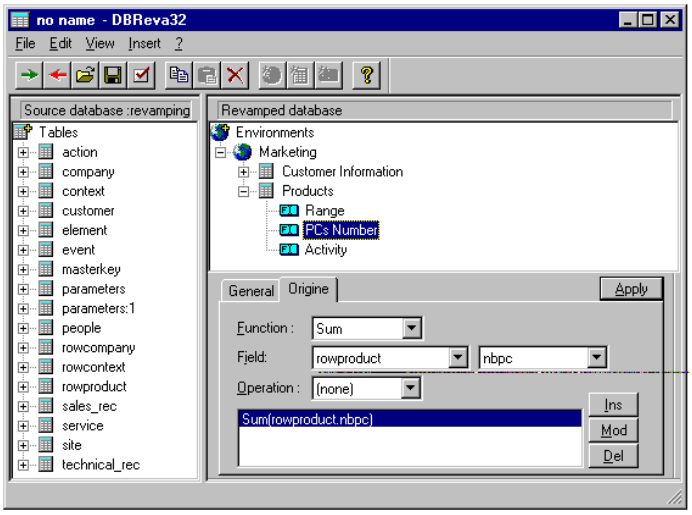
You can copy an existing field from a table in the real database directly into the virtual table; to do this:

- Use one of the methods described in the introduction (drag 'n drop, Copy/Paste, keyboard accelerator or toolbar button) to select the field in the table of the real database and place it in the virtual table of the redefined database.
- If you want to, you can change the name of the field and give it a description on the corresponding **General** tab (or else use the **F2** function-key to rename it).

New field

To define a new field in a virtual table:

- Select the option **Insert➔New Field** or choose **New Field** in the contextual menu for the table or use the toolbar button  .
- Enter a name and, optionally, a description on the **General** tab.
  Click the tab **Origin**. You can then:
  - Add a function to the field: select the function of your choice in the list box **Function**.
  - Add a value from an existing field in a real database to the new field or the function selected above: select the real table and field you want from the two Field list boxes.
  - Add an operation to the field selected above: choose the operator you want in the **Operation** list box.
- Then click the button **Ins** to add these options to the field definition.



The available functions are:
Sum, **Min**, **Max**, **Number**, **Average**, or **None**.

The available operations are:
+, - , * , / , or none. The operator + can be used to concatenate characters.

Example:
You have access to a real table "res_tab" which contains four fields res1, res2, res3 and res4 corresponding to the quarterly results in a particular year. You want to define the field "Result" in a table in your virtual database that contains the sum of the four real fields.
*On the tab **Origin**, for the field "Result":*
*Select the table "res_tab" and the field "res1" in the **Field** option's list boxes.*
*Select the operator + in the list box of the option **Operation**.*
*Click the button **Mod** to replace the default entry. The new entry is "res_tab.res1 +".*
*Next select the table "res_tab" and the field "res2" in the list boxes of the option **Field**.*
*Select the operator + from the **Operation** list box again.*
Click the button Ins to add the newly created entry "res_tab.res2 +".
Do the same for the field "res3".
*For the field "res4", select the operator **None** instead of +.*
*The **Result** field is finally defined from the following list:*

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

which means that the ***Result*** *field contains the sum of the four fields "res1"," res2", "res3" et "res4".*

You can change an item in the definition of a field using the button **Mod** (the highlighted element is replaced by the values selected above). Click the button Delete to delete the highlighted item.

After defining the field, click the button **Apply** for the new options to be taken into account.

As soon as a new virtual field has been defined, remember to define the joins between the table(s) used to create the virtual table, if there are any. Refer to the section "**Inter-table links**".

INTER-TABLE LINKS

The fields defined in a virtual table are obtained from one or more tables in the real database.

For each virtual table, it is essential to define the links between the real tables from which the constituent fields are extracted. The definition of these links make it possible to create the joins between the real tables when the end-user queries the virtual database. These links may be direct or indirect (i.e. links between one table and another or between more than one table and other tables).

Defining links

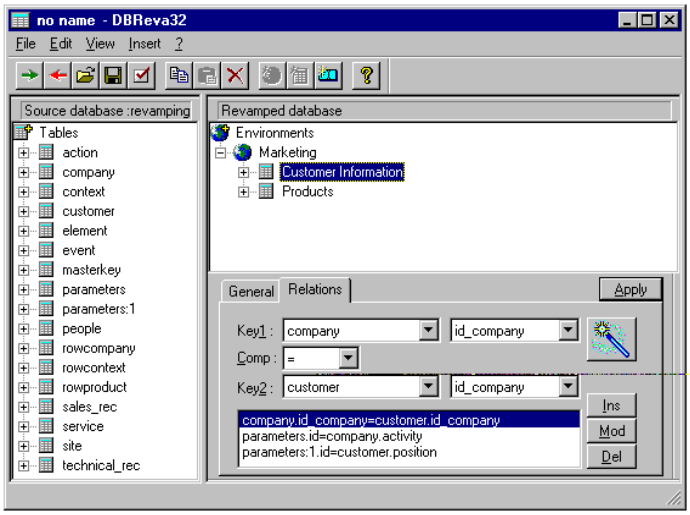
The simplest way is to define the links at the same time as the fields in the virtual table are defined. All the real tables used for defining fields must be linked directly or indirectly to the other real tables used by the virtual table.

To define links between real tables:

- Click the tab **Relations** for the virtual table in question.
- For each of the real tables (**Key1** for the first, **Key2** for the second), select its name and the field acting as the join with the other table using the list boxes.

Note:	The names of the two fields linking the tables may be different, even if they represent the same data.

- Select a comparison operator in the **Comp** list box.
- Click the button **Ins** to add the link to the list of links in the virtual table.



You can change a link with the button **Mod** after changing the values of the selected item. Click the button **Del** to delete the selected item.

Click the button **Apply** for the list of links thus defined to be validated.

Checking links

Tun DB Revamp provides a function for verifying the links defined by the administrator between the real tables used to construct the virtual table.

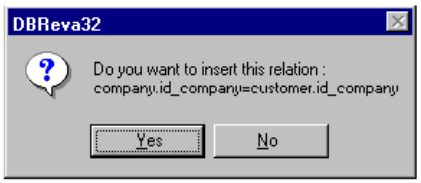
You can easily check for each of the virtual tables if the real tables used are linked and if the defined links form a coherent whole.

To do this, click the button  on the **Relations** tab.

Tun DB Revamp will then examine all the links defined by the administrator and detect possibly missing direct or indirect links that isolate particular tables from the rest.

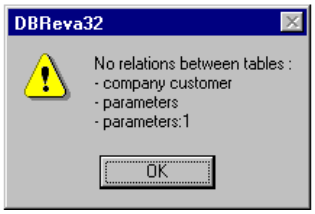
When a link between two tables is missing, **Tun DB Revamp** tries to link them using two fields of the same name.

If the two fields exist, **Tun DB Revamp** proposes to define the link as follows:



In most cases, the proposed link will be the right one. If, however, you think that the two fields proposed by **Tun DB Revamp** should not form the link between the tables, define the link manually as described in the section **Defining links**.

On the other hand, if two unlinked tables do not have fields of the same name, **Tun DB Revamp** displays a list of the unlinked tables:



In this case, define the link manually as described in the section **Defining links**.

EXPORTING A VIRTUAL DATABASE

The revamped database has to be exported from the PC to the server so it can be used by the different users or groups of users. The correspondence real "data source/environments" forms a virtual data source containing information on:

- the real database used
- the real ODBC driver used
- the revamping of the real database (environments, virtual tables and associated fields)

To export the virtual data source (real data source and revamped database) from the administrator's PC to the server, use the option **File➔Export Data Source** or else click the button  in the toolbar.

This creates or updates the three tables with the revamping information in the real database. Refer to the chapter "**Revamping in Tun SQL**".

VALIDATING AN ENVIRONMENT

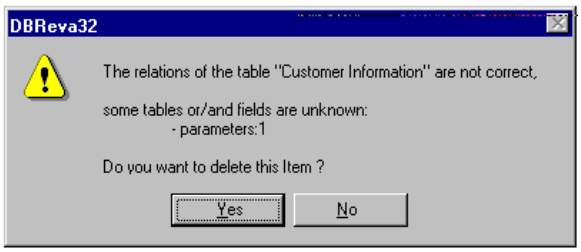
Tun DB Revamp includes a function for verifying the consistency between the contents of the environments created by the administrator and the contents of the real database used. This is especially useful when changes have been made in the structure of the real database (e.g. a field has been changed or deleted), that the administrator has not accounted for in the virtual database.

To use this function, the environment has to be validated before it is exported so as to avoid any possible inconsistency. Use the option **File➔Validate Environments** in the main menu or click the button  in the toolbar.

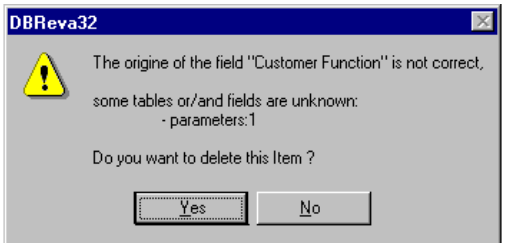
Example:
Take for the example the case where the table "parameters:1" and its fields have been copied to the environment "Marketing". Each field in the virtual table thus created has the table "parameters:1" as its point of origin.
If this logical table is consequently deleted in the real database, the fields in the virtual table will no longer have a point of origin. Similarly, all the tables with a reference to the table "parameters:1" in their relations will become inconsistent.

On a request for environment validation, **Tun DB Revamp**:

- indicates the virtual tables of which one or more relations refer to the deleted table and offers to delete them. In this case, it is preferable not to delete them but to delete the fields whose point of origin is in the deleted table.



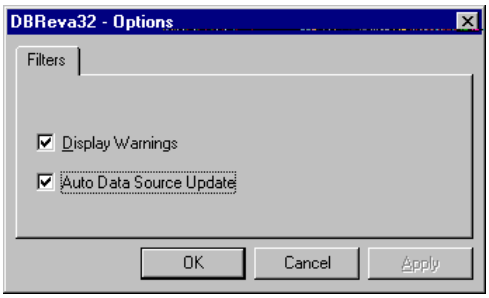
- indicates the virtual fields whose origin is in the deleted table and proposes to delete them.



If no inconsistency has been detected, the following window is displayed:

UPDATING A VIRTUAL DATA SOURCE

When you change the name of an environment, you can update the corresponding virtual data source. This functionality is automatic if you selected the check box **Automatic Data Source Update** in **View➔Options**.



If the option is selected, there are two possibilities:

- if the check box **Display Warnings** is selected, **Tun DB Revamp** will request confirmation whenever the automatic update is going to be performed.
- if this check box is not selected, the update will take place automatically without confirmation.

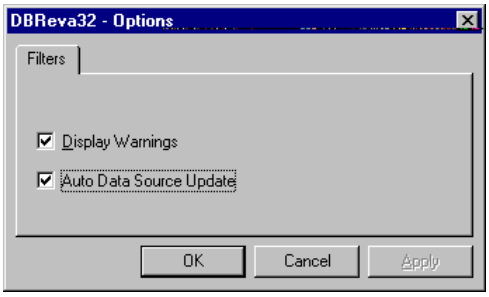
On the other hand, if the option **Automatic Data Source Update** is not selected, use the main menu option **Insert➔Create/Update Data Source** or the contextual menu option **Update Data Source** for the environment in question to update the corresponding data source.

If you want to cut the link temporarily between an environment and its data source, use the option **Insert➔Remove Data Source** or the contextual menu option **Remove Data Source**. The link can later be recreated using the option **Update Data Source**

DISPLAYING WARNINGS

A certain number of warnings can be displayed to the administrator while he is using **Tun DB Revamp**. These warnings supply information, for example, on inconsistencies which may arise during the revamping of the database or else warn of missing elements in the new structure.

Tun DB Revamp displays these warnings by default. However, you may cancel the display of message boxes by deselecting the check box Display Warnings (**View➔Options**).



LOCAL VIRTUAL DATA SOURCE MANAGEMENT

You may save and open locally the description of the revamped database. This might be useful if you do not want to export the revamped database immediately, or if you want to keep earlier versions. This description is saved in a file with the extension **.dbr**.

Saving locally

To save locally the description of a virtual database produced with **Tun DB Revamp**, use the option **File➔Save** (or **File➔Save As...** to save it under a different name), or else click the button  in the toolbar.

The path for the real data source is also saved, which means the virtual data source can later be exported without its destination having to be supplied.

Opening a local virtual data source

To open a virtual data source saved locally in a **.dbr** file, use the option **File➔Open** or click the button  in the toolbar.

Select the virtual data source you want to use.

If you have opened several virtual data sources simultaneously, you can change from one to another by clicking its name in the **File** menu.

If no data source has been opened, this menu option is called **Recent File** and is grayed.

INDEX