

# **ESKER** *Tun*<sup>®</sup> *Plus*

---

Tun SQL – Accesso a dati

Tun Plus 2009  
Issued May 2008

Copyright © 1989-2008 Esker S.A. All rights reserved.

© 1998-2002 The OpenSSL Project; © 1994-2003 Sun Microsystems, Inc.; © 1996 Wolfgang Platzer (wplatzer@iaik.tu-graz.ac.at); © 1995-1998 Eric Young (eay@cryptsoft.com). All rights reserved. Tun contains components which are derived in part from OpenSSH software. See the copyright.txt file on the Tun CD for additional copyright notices, conditions of use and disclaimers. Use and duplicate only in accordance with the terms of the Software License Agreement - Tun Products.

North and South American distributions of this manual are printed in the U.S.A. All other distributions are printed in France. Information in this document is subject to change without notice. No part of this document may be reproduced or transmitted in any form or by any means without the prior written consent of Esker S.A..



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

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

Esker Australia Pty Ltd. (Lane Cove - NSW) ♦ Tel: +61 (0)2 8596 5100 ♦ info@esker.com.au ♦ www.esker.com.au

Esker GmbH (München) ♦ Tel: +49 (0) 89 700 887 0 ♦ info@esker.de ♦ www.esker.de

Esker Italia SRL (Milano) ♦ Tel: +39 02 57 77 39 1 ♦ info@esker.it ♦ www.esker.it

Esker Ibérica, S.L. (Madrid) ♦ Tel: +34 91 552 9265 ♦ info@esker.es ♦ www.esker.es

Esker UK Ltd. (Derby) ♦ Tel: +44 1332 54 8181 ♦ info@esker.co.uk ♦ www.esker.co.uk

Esker, the Esker logo, Esker Pro, Extending the Reach of Information, Tun, and Tun Emul are trademarks, registered trademarks or service marks of Esker S.A. in the U.S., France and other countries.

The following are trademarks of their respective owners in the United States and other countries: Microsoft, Windows, Back-Office, MS-DOS, XENIX are registered trademarks of Microsoft Corp. Netscape and Netscape Navigator are registered trademarks of Netscape Communications Corp. IBM, AS/400, and AIX are registered trademarks of IBM Corp. SCO is a registered trademark of Caldera International, Inc. NetWare is a registered trademark of Novell, Inc. Sun, Sun Microsystems and Java are trademarks of Sun Microsystems, Inc. Oracle is a registered trademark of Oracle Corp. Informix is a registered trademark of Informix Software Inc. Sybase is a registered trademark of Sybase, Inc. Progress is a registered trademark of Progress Software Corp. All other trademarks mentioned are the property of their respective owners.

# INTRODUZIONE

**Tun SQL – Accesso a dati** è una suite di moduli applicativi e server che permette ai PC di operare in modalità client/server con database remoti (Informix, Oracle, Sybase, DB2, Progress e C-ISAM). **Tun SQL** utilizza l'architettura ODBC definita da Microsoft.

**Tun SQL** funziona sulle seguenti piattaforme: Windows 3.x, Windows 95, Windows 98, Windows NT 3.51, Windows NT 4.0, Windows 2000 e Citrix WinFrame, Citrix MetaFrame et Windows NT TSE.

**Tun SQL** fa parte della gamma di prodotti software **Tun** descritta di seguito:

	<b>Funzioni per Windows</b>	<b>Componenti in un ambiente multiutente</b>
<b>Esker TCP/IP Stack</b>	Stack di comunicazione TCP/IP per Windows 3.x (DLL)	n.d.
<b>Accesso a risorse di rete</b>	Applicazioni TCP/IP (NIS, NFS client e server, PING, reindirizzamento e condivisione delle stampanti, FTP client e server, TELNET, RSH client e server, TAR, WALL, TFTP, TIME)	Applicazioni TCP/IP (NIS, NFS client e server, PING, reindirizzamento e condivisione delle stampanti, FTP client e server, TELNET, VT320, RSH client, TAR, WALL)
<b>Accesso ai applicazioni</b>	Emulatore di terminale (asincrono, emulazione IBM3270 e IBM5250, stampante 3287/3812)	Emulatore di terminale (asincrono, emulazione IBM3270 e IBM5250, stampante 3287/3812)
<b>Accesso a dati</b>	Driver ODBC per TCP/IP in modalità client/server (DBMS Oracle, Informix, Sybase, DB2, Progress e C-ISAM) e tool per ridefinire ("revamping") i database	Driver ODBC per TCP/IP in modalità client/server (DBMS Oracle, Informix, Sybase, DB2, Progress e C-ISAM) e tool per ridefinire ("revamping") i database
<b>TCP/IP Network Services</b>	Browser NIS, reindirizzamento e condivisione delle stampanti	Reindirizzamento e condivisione delle stampanti

La maggior parte delle funzioni e delle procedure descritte nel presente manuale sono valide in ugual misura per Windows 3.x, Windows 95, Windows 98, Windows NT 3.51, Windows NT 4.0, Windows 2000 o Citrix/ Windows NT TSE. Alcune di esse tuttavia, valgono solo per alcune piattaforme; in questi casi il paragrafo o la sezione in questione saranno contrassegnati nel modo seguente:



Win 3.x

Windows 3.x



Win 98

Windows 95 e Windows 98



Win NT  
2000

Windows NT (Windows NT 3.51 e Windows NT 4.0, incluso ambiente multiutente salvo quanto altrimenti specificato) e Windows 2000



NT 4.0  
2000

Windows NT 4.0, incluso Citrix/Windows NT TSE salvo quanto altrimenti specificato



NT 3.51

Windows NT 3.51, incluso ambiente multiutente salvo quanto altrimenti specificato



Win 32

Windows a 32 bit (Windows 95, Windows 98, Windows NT 3.51 e Windows NT 4.0, incluso ambiente multiutente salvo quanto altrimenti specificato e Windows 2000)



Ambiente multiutente



Escluso ambiente multiutente

Con **Tun SQL** per Windows viene fornito anche **Tun PLUS**, che comprende tutti i moduli indicati sopra. Quando si installa **Tun PLUS** verrà proposto di installare anche **Tun SQL**. Con l'eccezione di **Tun PLUS** versione Multi-User Windows, è possibile installare **Tun SQL** indipendentemente da **Tun PLUS**.

**Nota**

In tutto il manuale, le caratteristiche di Windows 98 corrispondono a quelle di Windows 95.

# INDICE GENERALE

---

## PARTE 1 INTRODUZIONE E USO

CAPITOLO 1 - Introduzione a Tun SQL.....	1-9
Il sistema ODBC.....	1-9
Il modello Client/Server .....	1-11
ODBC e il modello SQL Client/Server .....	1-13
Tun SQL.....	1-13
CAPITOLO 2 - Configurazione e uso.....	2-17
Controllo del funzionamento di Tun SQL .....	2-17
Creazione di un database dimostrativo .....	2-21
Creazione di una origine dati.....	2-22
Copia del database dimostrativo .....	2-29
Creazione di una sorgente dati virtuale.....	2-31
Tabelle di conversione caratteri.....	2-33
CAPITOLO 3 - C-ISAM.....	3-37
Introduzione al C-ISAM.....	3-37
Utilizzo di sqltools.....	3-39

## PARTE 2 RIDEFINIZIONE DEL DATABASE

CAPITOLO 4 - Revamping .....	4-53
Database virtuali .....	4-53
Ridefinizione in Tun SQL .....	4-56
CAPITOLO 5 - Uso di Tun DB REVAMP.....	5-59
Opzioni generali .....	5-59
Importazione di ambienti di sorgenti di dati .....	5-61
Creazione di un ambiente .....	5-62
Creazione di una tabella virtuale .....	5-63
Creazione di un campo .....	5-64
Assegnazione ai campi di filtri .....	5-67
Collegamenti tra tabelle.....	5-69
Interrogazione di database reali e virtuali.....	5-72
Convalida di un ambiente .....	5-74
Esportazione di ambienti di sorgenti di dati .....	5-76
Aggiornamento di una sorgente di dati virtuale.....	5-76
Creazione di sorgenti di dati virtuali.....	5-77

Visualizzazione degli avvisi .....	5-78
Gestione locale delle sorgenti di dati ridefinite .....	5-79
Identificazione dei campi.....	5-80

**PARTE 3 APPENDICI**

APPENDICE A - Riferimento .....	A-85
---------------------------------	------

APPENDICE B - Comandi SQL utilizzati in C-ISAM .....	B-101
------------------------------------------------------	-------

Istruzioni principali.....	B-101
----------------------------	-------

Sintassi di comando SQL.....	B-102
------------------------------	-------

Tipi di dati .....	B-136
--------------------	-------

<b>INDICE ANALITICO.....</b>	<b>I-143</b>
------------------------------	--------------

**PARTE 1**  
**INTRODUZIONE E**  
**USO**





# INTRODUZIONE A TUN SQL

---

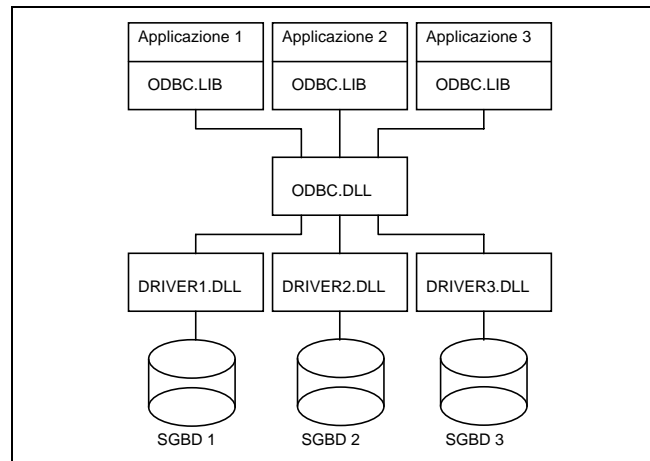
## Il sistema ODBC

Nella gestione dei sistemi database, il modello "embedded SQL" (SQL incorporato) viene utilizzato dai programmatori per interfacciare le applicazioni con i vari database. Il modello consente di inserire interrogazioni in linguaggio SQL all'interno di programmi scritti in COBOL e C e consente quindi una portabilità delle applicazioni su un ampio numero di piattaforme.

Il modello embedded SQL presenta tuttavia i seguenti lati negativi:

- Il numero degli embedded SQL è pari al numero dei sistemi DBMS presenti sul mercato. Un'applicazione che fa uso di SQL può dialogare solo con un DBMS specifico; per funzionare con altri, deve essere riscritta o quantomeno modificata. Se si desidera un'interfaccia destinata a funzionare con tutti i database presenti sul mercato il modello embedded SQL non risulta pratico.
- Il modello embedded SQL è relativamente poco sofisticato, presenta severe limitazioni e si rivela di uso poco pratico. In altre parole, non consente un pieno sfruttamento dei database al punto che a volte è preferibile utilizzare direttamente le API fornite dal sistema DBMS.

Per colmare queste carenze del modello embedded SQL Microsoft ha realizzato un nuovo modello basato su ODBC (Open Database Connectivity, connettività database aperta), basato a sua volta su architettura WOSA (Windows Open System Architecture, architettura Windows sistema aperto).



ODBC è un set di funzioni C predefinite per l'accesso e l'aggiornamento dei dati di un DBMS integrate in una DLL (Dynamic Link Library, libreria di collegamento dinamico) utilizzabile da qualsiasi applicazione Windows. La DLL (ODBC.DLL) analizza le richieste SQL, che vengono elaborate dai driver ODBC in modo da renderle compatibili con le API fornite dal DBMS utilizzato. Il driver ODBC consente di visualizzare l'interfaccia del DBMS e la rende disponibile alle applicazioni.

Microsoft fornisce la libreria ODBC.DLL e gli strumenti per utilizzarla, ma non fornisce i driver ODBC per tutti i DBMS sul mercato. Microsoft fornisce al contrario tutti i driver per le proprie applicazioni (Excel, Word, Access, ecc.).

I driver specifici possono essere forniti dai produttori dei singoli database o da terze parti, come nel caso di Esker.

Il massimo grado di interfacciamento del sistema ODBC consente anche alla più semplice applicazione Windows di accedere ai database. Il programmatore realizza e distribuisce l'applicativo senza doversi preoccupare di quale sistema database verrà utilizzato. Unico requisito è il driver che consente la comunicazione tra l'applicazione e il DBMS.

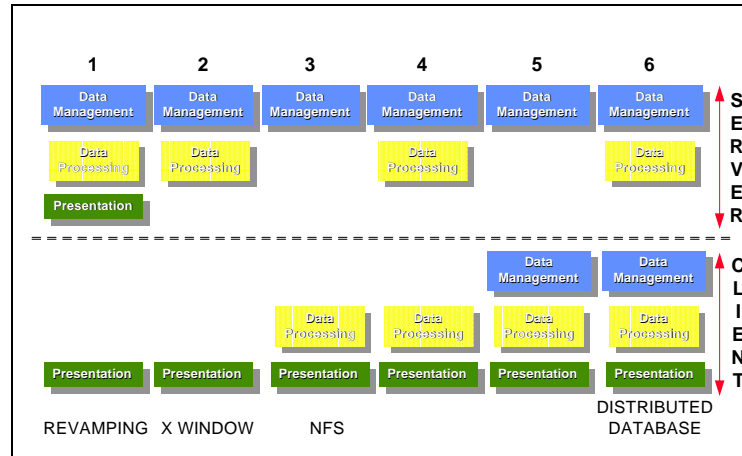
Per queste ragioni, il sistema ODBC si rivela particolarmente prezioso per l'uso di fogli elettronici, elaboratori di testi e tool di sviluppo che gestiscono informazioni residenti su DBMS non determinati a priori.



## Il modello Client/Server

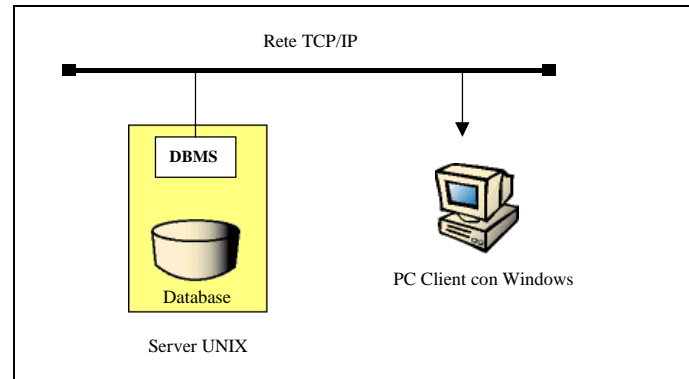
Il modello client/server, entrato nell'uso ormai da parecchi anni, contempla per ogni servizio fornito la collaborazione di due unità.

Il Gartner Group ha classificato sei tipi principali di applicazione client/server a seconda delle funzioni svolte dal client o dal server:



Le applicazioni che comportano un minor uso di risorse del client sono dette "revamping" o server X-WINDOWS. Le applicazioni che comportano un maggior uso di risorse del client sono quelle che utilizzano database distribuiti.

Tuttavia il termine "client/server" viene spesso utilizzato a proposito di applicazioni che utilizzano database distribuiti. Lo schema seguente illustra graficamente il funzionamento di questo modello:



Il PC client dispone di solito di un'interfaccia grafica per la consultazione dei dati, che risiedono invece su server UNIX e sono gestiti da un RDBMS (Relational Database Management System, sistema di gestione database relazionale). Per la consultazione o l'aggiornamento dei dati, il client invia una richiesta SQL al server, che la esegue e restituisce la risposta via rete.

Il sistema presenta i seguenti vantaggi:

- L'utente finale dispone di un'interfaccia intuitiva integrata all'ambiente Windows.
- Il PC non è "dedicato", cioè può essere utilizzato anche per scopi diversi da quelli di ufficio (office automation, calcoli, personali, ecc.).
- Più utenti possono accedere contemporaneamente alle informazioni sul server utilizzando ciascuno il proprio PC.
- Il server viene sgravato da parte dei task e può concentrare le proprie risorse informatiche per la fornitura dei dati. Il carico di elaborazione viene ripartito tra i client e il server.
- La rete serve solo al trasporto dei dati e non viene così sovraccaricata dall'esecuzione dei programmi client di interrogazione.



Il modello SQL client/server descritto è una versione moderna del modello transazionale quale è ancora utilizzato negli ambienti con mainframe e terminali sincroni.

## ODBC e il modello SQL Client/Server

Il modello ODBC semplifica l'accesso a tutti i sistemi di gestione dei dati, ivi inclusi i database remoti. Un'applicazione Windows può, tramite ODBC, utilizzare i dati indipendentemente dalla loro origine e, per converso, applicazioni come **Excel**, **Word**, **Access** o altre residenti su PC remoti possono accedere ai medesimi dati aziendali residenti su sistema centrale.

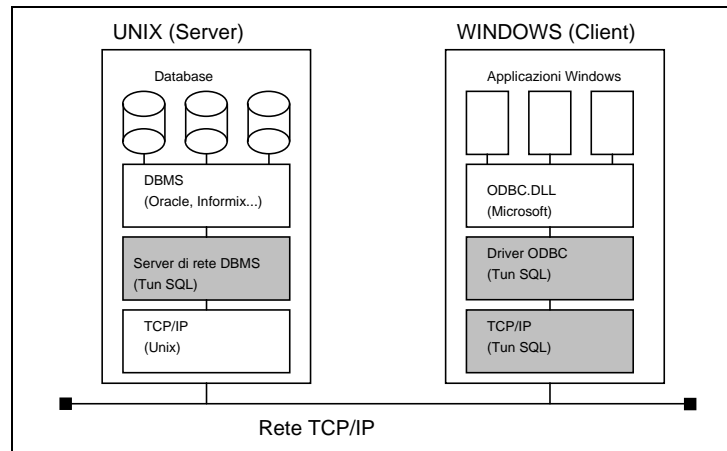
Ciò comunque non significa che la realizzazione di un sistema basato su questo modello sia cosa facile. Esso infatti richiede, oltre ad una rete di PC locale, un server UNIX e un sistema DBMS, le seguenti componenti:

- La porzione server del DBMS (Informix Net, Sql Net, ecc.).
- La porzione client per il PC (Informix Net PC, Sql Net PC, ecc.).
- Il driver ODBC appropriato.
- Stack TCP/IP compatibile **Winsock** per il PC.

I produttori dei sistemi DBMS forniscono in ogni caso le prime due componenti. Altrettanto non si può dire per il driver, mentre è da escludere che forniscano gli stack TCP/IP. È quindi necessario procurarsi le ultime due componenti facendo particolare attenzione che siano compatibili con la configurazione in uso.

## Tun SQL

**Tun SQL** è stato realizzato per risolvere questo problema in modo definitivo in quanto integra in un unico prodotto tutti i componenti menzionati, oltre ad una sofisticata funzione per la ridefinizione ("revamping") dei database e un driver ODBC virtuale per l'accesso al database ridefinito. **Tun SQL** comprende inoltre i componenti di rete del DBMS (client e server). Ciò costituisce un grande vantaggio soprattutto per l'installazione di un ambiente che comprende più PC.



Le caratteristiche principali del pacchetto sono descritte di seguito.

### ➤ Driver ODBC unico per la maggior parte dei DBMS sul mercato

Per limitare il numero dei prodotti software richiesti per il client PC, **Tun SQL** comprende un unico driver ODBC per l'accesso ai seguenti DBMS:

- Oracle versione 7.
- Informix versione 5 e 7.
- Sybase versione 10.
- DB2 versione 2.
- Progress versione 6, 7 e 8.
- C-ISAM versione 4 a 7.

### ➤ Ridefinizione del database

Grazie all'uso di un'applicazione semplice da utilizzare, **Tun SQL** consente di ridefinire le tabelle di un database in modo da facilitarne l'accesso da parte dell'utente finale. Ciò è possibile tramite una riorganizzazione personalizzata delle tabelle, la modifica dei nomi di tabelle e campi e alcune funzioni preimpostate.



### ➤ Driver ODBC virtuale per database ridefiniti

Per poter utilizzare un database ridefinito, **Tun SQL** comprende un driver ODBC virtuale che traduce le richieste inviate alle tabelle virtuali in richieste gestibili da un normale driver ODBC.

### ➤ Porzione server del DBMS inclusa in Tun SQL

Ogni pacchetto contiene la porzione server per i vari DBMS e i seguenti ambienti Unix:

- 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.

Ciò risparmia il costo della porzione server del DBMS.

### ➤ Stack TCP/IP forniti di serie



Come anche gli altri programmi della stessa serie, **Tun SQL** viene fornito con **Esker TCP/IP Stack**, ottimizzati in modo da fornire le migliori prestazioni. Oltre a questo, il cliente non deve preoccuparsi di acquistare altrove gli stack.

### ➤ Installazione e gestione semplici

**Tun SQL** prevede una procedura di installazione semplice per i sistemi server UNIX e per i client Windows e fornisce una documentazione completa del software. Oltre al driver ODBC, sono disponibili due applicazioni Windows per il test e l'implementazione del modello client/server:

- **Tun DB Show**, per il test della connessione client/server da una terminazione all'altra. L'applicazione è in grado di ottenere dal server UNIX l'elenco dei DBMS e dei database installati.
- **Tun DB Script** avvia file batch SQL in Windows che creano dei database sui DBMS remoti.

Oltre ai sistemi di protezione propri di Unix e di ciascun DBMS, **Tun SQL** dispone di un sistema in grado di impedire ad alcune applicazioni Windows l'accesso a database riservati.



L'integrazione del servizio NIS (Network Information Service, servizio informativo di rete) all'interno di **Tun SQL** consente una gestione centralizzata delle risorse di rete e ne facilita l'accesso da parte degli utenti remoti. Per ulteriori informazioni su NIS, consultare il manuale **TCP/IP Network Services**.

### ➤ **Compatibilità del driver Tun SQL**

Il driver di **Tun SQL** supporta tutte le funzioni del livello 1 ed alcune del livello 2. Con il driver viene fornita la "Microsoft ODBC Cursor Library", sufficiente per un gran numero di applicazioni nonostante supporti solo cursori statici e "solo avanti".





## CONFIGURAZIONE E USO

---

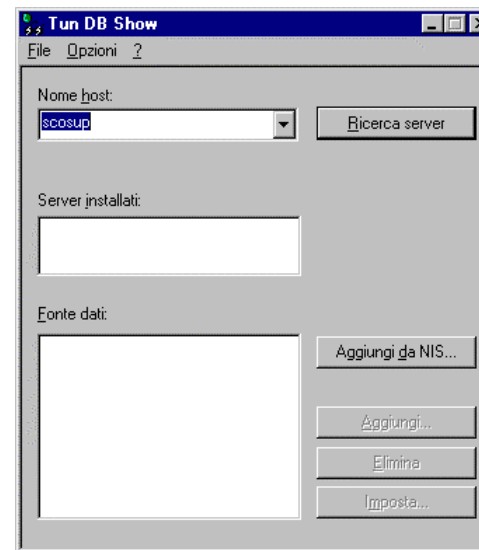
### Controllo del funzionamento di Tun SQL

#### ➤ Eseguire Tun DB Show

Al termine dell'installazione e della configurazione di **Tun SQL** su Windows e su UNIX è necessario controllarne il corretto funzionamento eseguendo il programma **Tun DB Show**.



Avviare il programma facendo clic sull'icona **Tun DB Show** nei gruppi **Data Access** (menu **Avvio, Programmi, Esker Tun** in Windows 95/98/2000 e Windows NT).



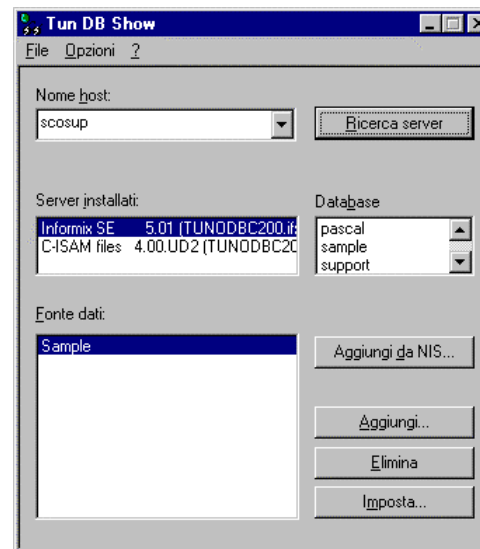
Il programma verifica quanti server **Tun SQL UNIX** sono installati su un host UNIX. Indicare l'host nel campo **Nome host**, oppure selezionare il server dall'elenco a discesa (questo elenco mostra i server che sono dichiarati nel file **hosts** e sul server NIS).



Per ulteriori informazioni sulla configurazione di **Tun NIS**, consultare il manuale **TCP/IP Network Services** o **Tun NET**.

Premere **Ricerca server**.

I programmi server correttamente installati sul sistema remoto compariranno nell'elenco **Server installati**.



Ciascuna riga indica:

- Il nome del sistema DBMS interfacciato con il server **Tun SQL UNIX**.
- La versione del DBMS.
- Il nome dell'eseguibile contenente il server **Tun SQL UNIX** (**tunodbc200.ora**, ad esempio).



**Nota:**

Con alcuni DBMS (come Informix On-Line, ad esempio), la selezione del server nell'elenco visualizza l'elenco dei database gestiti dal DBMS nella colonna Database.

Se l'elenco non contiene server **Tun SQL**, ciò significa che si è verificato un problema durante l'installazione. In questo caso le operazioni e le verifiche descritte nei capitoli precedenti devono essere ripetute.

## ➤ Parametri

La lista dei servizi può essere utilizzata come segue:

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

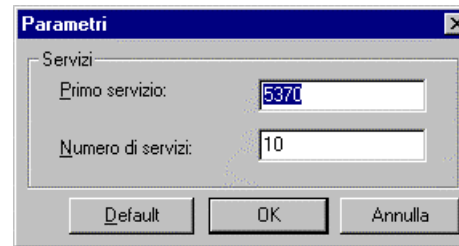
Utilizzare il menu **Opzioni** in **Tun DB Show** per:

- Gestire i servizi.
- Immettere le impostazioni del server proxy.

### Gestione servizi

Un numero di servizio a partire da 5370 viene associato con ogni processo server **Tun SQL**. Per un corretto funzionamento di **Tun SQL**, è necessario definire il Primo Servizio e poi il Numero dei Servizi richiesti. Questo consente a **Tun SQL** di identificare i diversi sistemi database accessibili al server **Tun SQL**. Il valore predefinito per il **Primo Servizio** è 5370 e quello per il Numero di Servizi è 5.

Selezionare **Opzioni**→**Parametri...** dal menu principale. Appare la seguente finestra di dialogo:

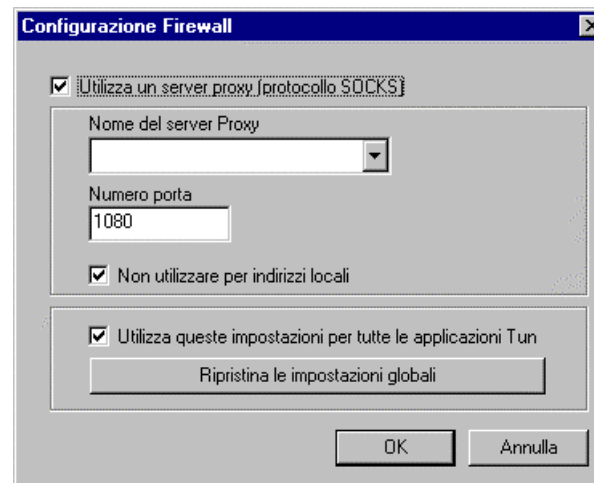


### Utilizzo di un server proxy firewall.

Configurando un server proxy in **Tun DB Revamp**, l'accesso ad un server esterno avviene tramite una macchina gateway proxy.

Per impostare i parametri del firewall (indirizzo IP, numero porta, ecc.), selezionare **Opzioni**→**Firewall** dal menu principale.

Appare la seguente finestra di dialogo:



Selezionare la casella **Utilizza server proxy**.



Immettere il nome o l'indirizzo IP del server. Immettere un nome soltanto se si utilizza un DNS. E' anche possibile selezionarne uno dalla lista a tendina (fare clic sulla freccia in giù alla destra del campo). La lista contiene i nomi dei server listati nella tabella server (hosts) e sul server NIS (le risorse NIS hanno icone gialle).

Immettere anche il numero porta SOCKS (normalmente il valore predefinito è 1080).

Per evitare l'utilizzo del firewall per le connessioni locali, selezionare **Non utilizzare per indirizzi locali**.

La configurazione del firewall può essere resa valida per tutte le applicazioni **Tun**: per fare questo, selezionare casella **Utilizza queste impostazioni per tutte le applicazioni Tun**. Per ripristinare la configurazione generale per tutte le applicazioni **Tun** in uso (per esempio, dopo aver utilizzato una configurazione **Tun NFS**), fare clic su **Ripristina le impostazioni generali**.

## Creazione di un database dimostrativo

Per familiarizzare gli utenti con le funzionalità ODBC in ambiente client/server, **Tun SQL** viene fornito completo di esempi pratici.

Per utilizzare gli esempi forniti è necessario creare un database per uno dei sistemi DBMS installati. Data la complessità dell'operazione per alcuni DBMS (ad esempio Oracle), è possibile utilizzare database preesistenti a patto che non contengano dati importanti. Per compatibilità con le istruzioni riportate in seguito, è consigliabile chiamare il nuovo database **tunsqldemo**.

## Creazione di una origine dati

### ➤ Origine dati

Per utilizzare un driver e un database specifici, le applicazioni compatibili ODBC devono poter riconoscere una Origine dati (Data Source). Il termine **Origine dati** si riferisce al driver ODBC utilizzato (ad esempio, **tunodb32.dll**) e ai parametri di funzionamento, ovvero:

- Nome o indirizzo IP dell'host UNIX remoto.
- Tipo di DBMS (Oracle, Informix, Sybase, DB2, Progress, C-ISAM).
- Nome del database.
- Note.
- Informazioni aggiuntive.

### ➤ Creazione di una Origine dati

Gli esempi forniti con **Tun SQL** utilizzano la stessa sorgente di dati (eccetto gli esempi relativi al revamping del database). È necessario creare questa sorgente dati prima di poter utilizzare gli esempi (seguire le istruzioni nel manuale "**Iniziare con Tun**").

Per creare una sorgente dati è possibile utilizzare:

- **Tun DB Show**.
- **Amministratore Sorgenti ODBC** (utilità Windows).

#### Creazione di sorgenti dati con Tun DB Show

Eeguire **Tun DB Show** ed effettuare quanto segue:

- Immettere il nome o l'indirizzo IP del server su cui è installato il database per il quale si desidera creare la sorgente dati.
- Fare Clic **Ricerca Server** per visualizzare i moduli server **Tun SQL** installati su host UNIX.
- Selezionare il modulo server DBMS per il database di cui si desidera creare la sorgente dati.
- Fare Clic su **Aggiungi...**





Se è installato **Tun NIS** su PC e l'amministratore di rete ha configurato le tabelle NIS, è possibile utilizzare il pulsante **Aggiungi da NIS...** per accedere alle sorgenti di dati in rete. Fare riferimento al manuale **Servizi di rete TCP/IP** per le istruzioni sulla configurazione di **Tun NIS**.

►► Leggere la sezione "**Configurazione della sorgente di dati**".

### **Creazione di una sorgente di dati con l'Amministratore Sorgenti ODBC**

Aprire il **Pannello di Controllo** e fare clic sull'icona ODBC (32bit ODBC in Windows 95). Fare clic sul pulsante **Aggiungi** nella finestra di dialogo che appare.

Selezionare il driver ODBC **Tun32 Driver**.

### **Configurazione della sorgente di dati**

Fare clic sul pulsante **Aggiungi** in **Tun DB Show** oppure in **Amministratore Sorgenti ODBC** per visualizzare la finestra di dialogo:

The screenshot shows the 'Tun SQL Setup' dialog box with the following fields and values:


- Nome fonte dati: TunSqlDemoOra
- Descrizione: Tun SQL Demo
- Nome host: everest
- Nome servizio: TUNODBC200.ora
- Database: tunsqldemo
- Login: (empty)
- Password: (empty)

Buttons at the bottom: NIS Import..., OK, Annulla

## ► Generale

Di seguito la descrizione dei campi della finestra.

### Nome fonte dati (Origine dati)

L'icona  indica il campo contenente il nome dell'origine dati utilizzata dalle applicazioni compatibili ODBC. Per poter effettuare la connessione ai database di esempio le origini dati devono tassativamente avere i nomi seguenti:

- **TunSqlDemoIfx** per Informix On-Line
- **TunSqlDemoIse** per Informix SE
- **TunSqlDemoOra** per Oracle
- **TunSqlDemoSyb** per Sybase

### Descrizione

Note relative all'Origine dati.

### Nome host

Indirizzo IP o nome dell'host contenente il database da utilizzare.

### Nome servizio

Il nome del programma server **Tun SQL** associato al DBMS contenente il database, ad esempio **tunodbc200.ora**.

Se vengono utilizzati stack TCP/IP diversi di **TCP/IP Stack**, aggiungere le istruzioni seguenti al file SERVICES del software TCP/IP:

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

Gli altri servizi sono i seguenti (non utilizzati nell'esempio):

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





## Database

Immettere il nome del database che si desidera utilizzare. Per utilizzare il database di esempio, immettere **tunsqldemo** (il database di esempio fornito con **Tun SQL**).

## Nome utente

Indica il nome dell'utente autorizzato all'accesso del database.

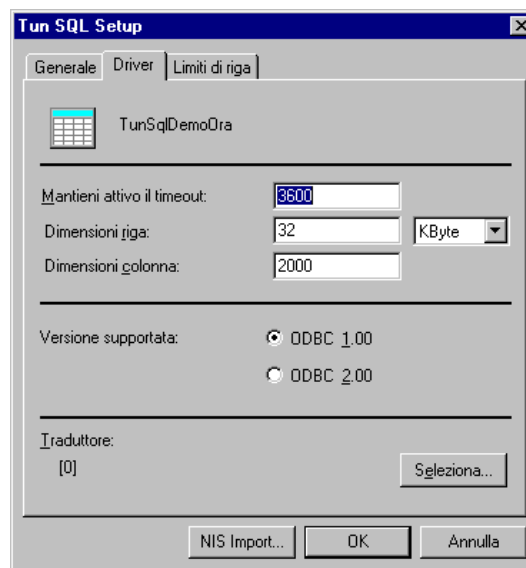
## Password

Indicare la password associata all'utente.

## NIS

Se è installato **Tun NIS** sul PC e l'amministratore di rete ha configurato le tabelle NIS, è possibile fare clic sul pulsante **Importa da NIS...** per accedere alle sorgenti dati in rete. Fare riferimento al manuale **Servizi di rete TCP/IP** (se non si ha **Tun NET**) per le istruzioni sulla configurazione di **Tun NIS**, oppure al capitolo "**Il Navigatore NIS**" nel manuale **Tun NET** o **TCP/IP Network Services**.

Fare clic sulla scheda **Driver** per visualizzare la finestra di configurazione del driver ODBC.



### **Mantieni attivo il timeout**

Il PC è un sistema soggetto a errori hardware e software e per questo il programma server **Tun SQL** controlla che il PC stia funzionando correttamente inviando pacchetti di dati ad intervalli regolari. Se il PC non risponde entro **n** secondi, il processo viene interrotto. Il campo imposta il tempo **n** (predefinito: 1 ora).

### **Dimensioni riga**

Indica le dimensioni dei pacchetti di dati estratti da una tabella durante un'operazione SQL di "select". Il valore può essere espresso in KB o numero di righe.

Se il valore è uguale a 1 significa che viene estratto un pacchetto TCP per riga. Se il valore è uguale a 100, le righe vengono riunite in pacchetti di 100 fino a raggiungere il numero di righe effettivamente recuperate. Tale valore consente di ottimizzare le trasmissioni all'interno della rete. Il valore ottimale è compreso tra 50 e 150 (predefinito: 32 KB).

### **Dimensioni colonna**

L'unità di suddivisione per il recupero di colonne di dati molto larghe, cioè contenenti molti dati. Un valore basso comporterà un sensibile aumento delle trasmissioni di dati via rete.

### **Versione supportata**

Al momento esistono due versioni di ODBC, rispettivamente 1.00 e 2.00. Alcune applicazioni, come Microsoft Access, funzionano solo con la prima versione. Il driver ODBC fornito da **Tun SQL**, compatibile con la versione 2.00, è in grado di emulare la versione 1.00. Attivare la casella indicando il grado di compatibilità ODBC richiesta dall'applicazione utilizzata.

### **Traduttore**

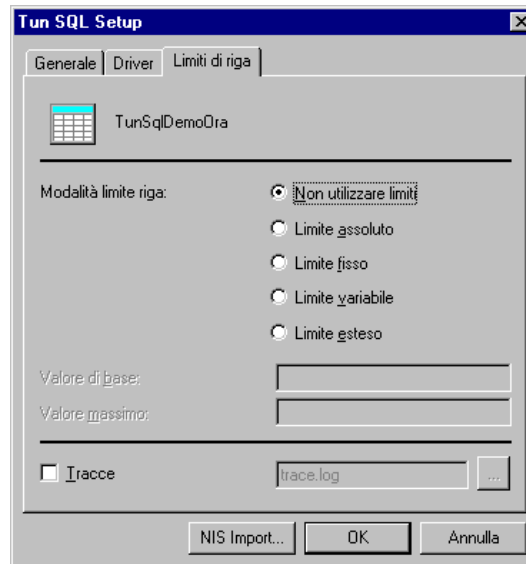
Date le differenze di gestione dei caratteri estesi in Windows (CP850) e UNIX (ISO8859), può essere necessario attivare delle tabelle di conversione dei caratteri per il driver ODBC. Il pulsante **Selezione** attiva le tabelle di conversione scelte. Non è necessario utilizzare queste tabelle per il database dimostrativo in quanto non contiene caratteri estesi.



**Nota:**

Le tabelle di conversione possono essere create o modificate con il programma **Tun DB Map**.

Fare clic sulla scheda **Limiti di riga** per visualizzare la finestra di dialogo seguente:

**Modalità limite riga**

Alcune applicazioni di office automation consentono all'utente di creare richieste SQL particolari. Altre applicazioni leggono una tabella intera prima di visualizzarla. Ciò non rappresenta un problema per tabelle di dimensioni contenute, ma può essere un problema insormontabile nel caso di grandi quantità di dati residenti su database remoti.

In questo caso infatti il numero di transazioni di rete aumenta enormemente e la memoria del PC potrebbe essere insufficiente a contenere i dati ricevuti, con conseguente necessità di riavviare il PC dopo un'operazione di "select". Per ovviare a questo problema, è possibile impostare dei limiti applicati dal driver ODBC con il pulsante **Limiti di riga...**

Vi sono cinque tipi di limite:

<b>Non utilizzare limiti</b>	Il driver non imposta alcun limite.
<b>Limite assoluto</b>	Il driver non carica più di <b>n</b> righe per operazione <b>select</b> . L'utente non viene avvisato.
<b>Limite fisso</b>	Il driver non carica più di <b>n</b> righe per operazione <b>select</b> . L'utente viene informato con un messaggio.
<b>Limite variabile</b>	Il driver non carica più di <b>n</b> righe per operazione <b>select</b> . Un messaggio comunque propone di caricarne altre senza superare il limite indicato da <b>Massimo</b> .
<b>Limite esteso</b>	Il driver non carica più di <b>n</b> righe per operazione <b>select</b> . Un messaggio comunque propone di caricarne altre senza un limite Massimo. Il messaggio visualizzato ha solo valore informativo.

### Tracce

Attivando la casella **Tracce** è possibile tracciare le query SQL in un file **.log** consultabile in un secondo momento. Tale opzione consente di monitorare l'esecuzione del driver ODBC durante l'invio di una query SQL.

Fare clic sul pulsante ... per scegliere la directory in cui collocare questo file.

### NIS



Se è installato **Tun NIS** su PC e l'amministratore di rete ha configurato le tabelle NIS, è possibile utilizzare il pulsante **Aggiungi da NIS...** per accedere alle sorgenti di dati in rete. Fare riferimento al manuale **Servizi di rete TCP/IP** per le istruzioni sulla configurazione di **Tun NIS**.



**Nota:**

In precedenza è stato consigliato di chiamare l'Origine dati **tunsqldemoXXX** in quanto gli esempi forniti fanno riferimento a questo nome.

Per utilizzare **Tun SQL** con altre applicazioni o database, è necessario di volta in volta creare l'Origine dati. In generale, ci devono essere Origini dati specifiche per ciascuna applicazione e ciascun database utilizzati.

È possibile creare l'Origine dati con l'applicazione ODBC nel **Pannello di configurazione**.

## Copia del database dimostrativo

Il pacchetto **Tun SQL** comprende un database dimostrativo utilizzabile con i programmi acclusi. Il database va copiato dal PC all'Origine dati **tunsqldemoXXX** creata nella sezione precedente.

**Nota:**

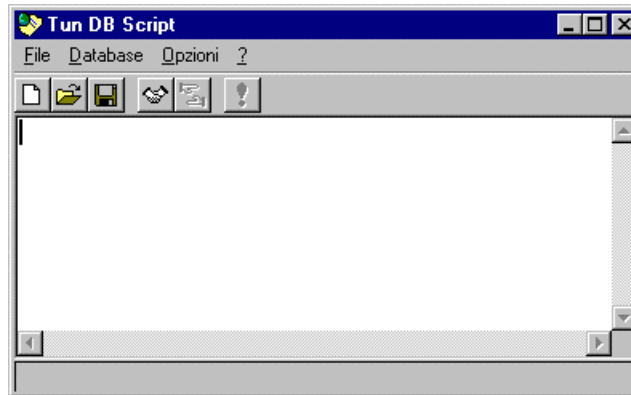
La variabile **XXX** può assumere uno dei valori seguenti:

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




Per effettuare la copia, avviare il programma facendo clic sull'icona **Tun DB Script** nei gruppi **Data Access** (menu **Avvio**, **Programmi**, **Esker Tun** in Windows 95/98/2000 e Windows NT).


Verrà aperta la finestra:



### ➤ Caricamento del file batch SQL per la creazione del database


Caricare il database scelto con il comando **File**→**Apri** o facendo clic su . Il file `\Demo\Db\Xxxcreat.sql` è contenuto nella directory di installazione di **Tun SQL**.

### ➤ Connessione all'Origine dati

Prima di avviare il file batch SQL, connettersi all'Origine dati facendo clic su  o con il comando **Database**→**Collega**. Verrà visualizzata una finestra di dialogo in cui indicare il nome dell'Origine dati. Il file batch viene eseguito solo a connessione riuscita.


Per caricare il database dimostrativo, selezionare l'Origine dati **TunSqlDemoXXX**, già definita in precedenza.

### ➤ Esecuzione

Eeguire il file batch SQL con il comando **Database**→**Esegui** o facendo clic su . **Tun DB Script** invia in sequenza i comandi SQL al database scelto. In caso di errore, lo script viene interrotto e viene visualizzato un messaggio di errore.



## ➤ Scollegamento dall'Origine dati

Al termine dell'esecuzione è necessario chiudere il collegamento con l'Origine dati facendo clic su  o con il comando **Database** ➔ **Scollega**. La connessione viene chiusa anche terminando l'applicazione.

**Nota:**

Sebbene la funzione principale di **Tun DB Script** sia di caricare il database dimostrativo **Tun SQL**, esso può eseguire liste di comandi SQL per creare altri database e per aggiornare o cancellare tabelle di grandi dimensioni.

## Creazione di una sorgente dati virtuale

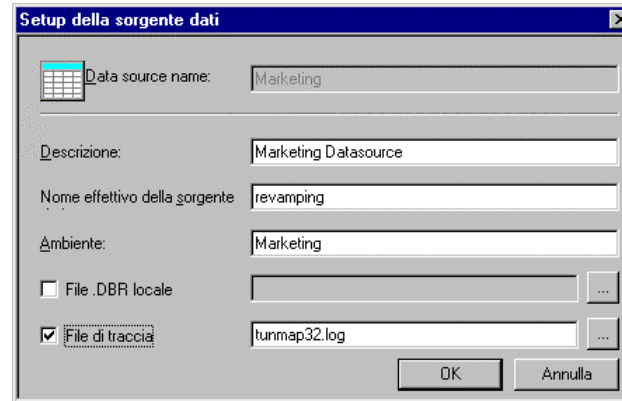
**Tun DB Revamp** può collegare tabelle virtuali, adattate all'ambiente operativo dell'utente finale, ad un database reale. Fare riferimento alla sezione "**Tun DB Revamp**" per ulteriori informazioni su questa applicazione.

Nella creazione di un database virtuale (revamped), è possibile altresì creare sorgenti dati per questo, esattamente come se si trattasse di un database reale. Gli utenti possono utilizzare il driver ODBC virtuale fornito con **Tun SQL** per accedere al database virtuale creato appositamente per loro.

Una sorgente dati virtuale può essere considerata come associazione di una sorgente reale ed un ambiente.

Per creare un sorgente dati virtuale, accedere alla configurazione della sorgente dati virtuale nell'**Amministratore Sorgenti ODBC**: scegliere il driver **Tunmap32**. Fare riferimento a "**Creazione di sorgenti di dati**".

Appare la seguente finestra di dialogo:



#### **Data source name**

Immettere il nome della sorgente revamped.


#### **Descrizione**

Immettere una descrizione per identificare la sorgente dati.

#### **Nome effettivo della sorgente**

Immettere il nome della sorgente dati del database reale da cui dipende il database virtuale.

#### **Ambiente**

Immettere il nome dell'ambiente per cui si sta creando una sorgente dati virtuale. Un ambiente è un insieme di tabelle virtuali. Ogni database revamped può avere uno o più ambienti. Fare clic sul pulsante  per scegliere l'ambiente dalla lista definita nel database.

#### **File .DBR locale**

Invece di immettere un nome di sorgente dati reale, è possibile selezionare l'ambiente da un file ".dbr" locale. Fare riferimento alla sezione "**Tun DB Revamp**" per informazioni sull'uso di questo tipo di file.





Selezionare la casella di controllo **File .DBR locale**. Immettere il percorso completo del file o fare clic sul pulsante sfoglia  per selezionare il file ".dbr". Quindi scegliere l'ambiente (campo **Ambiente**).

#### **File di traccia**

Per eseguire una traccia delle interrogazioni SQL, selezionare la casella di controllo **File di traccia**. La registrazione viene salvata in un file log consultabile con un editore di testo. Ciò permette di visualizzare cosa succede quando viene inoltrata una query SQL al driver ODBC.

Fare clic sul pulsante sfoglia  per scegliere la directory in cui si desidera salvare il file log e digitare un nome file.

## **Tabelle di conversione caratteri**

**Si può ignorare questo paragrafo al momento della prima lettura.**

### ➤ **Sistemi di notazione sui vari sistemi**

Mentre il set di caratteri ASCII (codici da 0 a 127) comprende tutti i caratteri dell'alfabeto inglese, esso non comprende i caratteri accentati o con altri segni diacritici di lingue come il francese, il tedesco e l'italiano. Sebbene siano state definite norme precise a questo scopo, esse tuttavia non risultano implementate in tutti i sistemi informatici.

Siccome **Tun SQL** prevede l'accesso a DBMS UNIX da PC (due sistemi con sistemi di codifica dei caratteri diversi), esso è in grado di gestire in entrambi i sensi la conversione dei caratteri da un sistema all'altro. In questo modo i caratteri speciali vengono interpretati correttamente da un sistema all'altro.

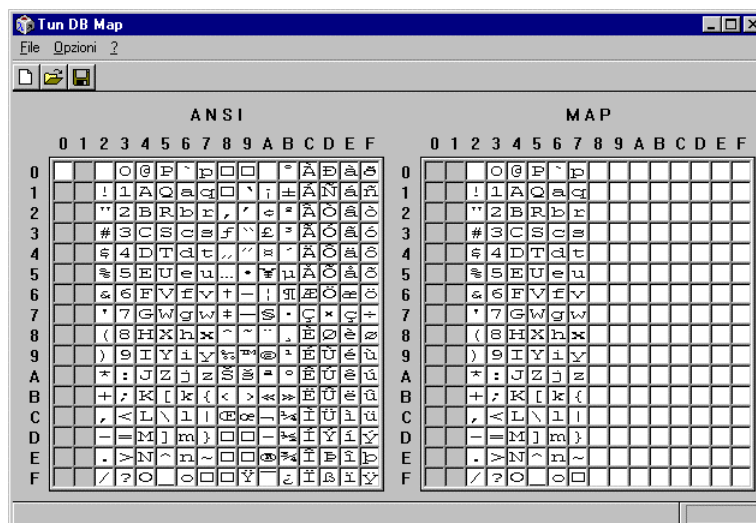
### ➤ **Creazione di tabelle di conversione**

Il sistema di gestione dei caratteri speciali implementato in **Tun SQL** utilizza tabelle di conversione che possono essere create o aggiornate con il programma **DBMAP**.



Avviare il programma facendo clic sull'icona **Tun DB Map** nei gruppi **Data Access** (menu **Avvio**, **Programmi**, **Escher Tun** in Windows 95/98/2000 e Windows NT).

Verrà visualizzata la finestra seguente:



La tabella a sinistra riporta i caratteri utilizzati sul PC (ASCII e ASCII esteso), mentre quella a destra riporta gli stessi caratteri ma disposti secondo la notazione utilizzata dal DBMS UNIX. Le prime 128 posizioni coincidono in entrambi i sistemi e vengono quindi visualizzate automaticamente.

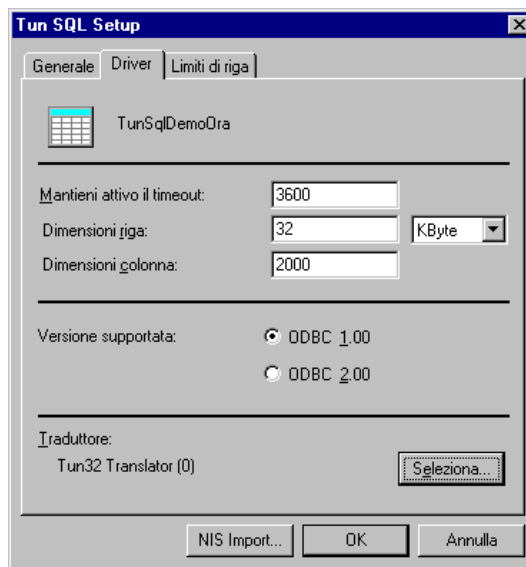
Per assegnare un carattere ad una posizione nella tabella destra, selezionare il carattere nella tabella sinistra e trascinarlo nella posizione desiderata a destra.

È possibile creare più tabelle di conversione con il comando **File** e salvarle con estensione **".ttt"**.



## ➤ Implementazione delle tabelle di conversione

Per renderle attive, le tabelle di conversione devono essere associate con un'Origine dati tramite la finestra di dialogo appropriata. Indicare le informazioni nella sezione **Traduttore**.



oppure premere **Seleziona...** per selezionare il convertitore ODBC da utilizzare.



## C-ISAM

---

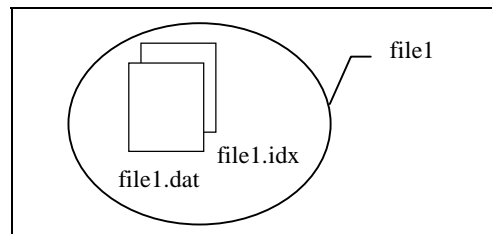
### Introduzione al C-ISAM

#### ➤ Il file system C-ISAM

C-ISAM (Indexed Sequential Access Method) è una libreria di funzioni C sviluppata da Informix. Permette la gestione di file sequenziali ad indici (creazione di file e operazioni di inserimento, cancellazione e lettura). C-ISAM include altre caratteristiche tipo il blocco di file (locking) ed il supporto transazionale che assicura l'integrità nei dati. Queste caratteristiche garantiscono che i dati siano accessibili, validi e utilizzati correttamente.

C-ISAM utilizza tipi di dati simili a quelli utilizzati in C. Dal momento che C-ISAM implementa questi tipi indipendentemente dal sistema UNIX utilizzato, il modo in cui i dati sono memorizzati può differire dal modo in cui i dati vengono rappresentati al momento dell'esecuzione. C-ISAM include funzioni per la conversione del formato dati "run-time" in formato dati memorizzabile

Un file C-ISAM è in realtà la combinazione di due file: uno contiene i dati (file.dat) e l'altro un indice per trovare i dati nel file dati (file.idx). I due file sono sempre utilizzati insieme come un file logico C-ISAM.



## ➤ RDBMS e C-ISAM

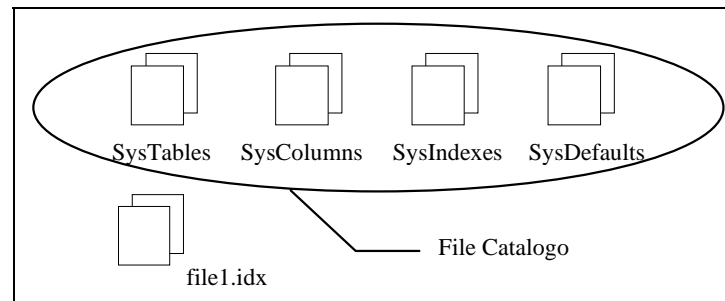
Siccome i file C-ISAM utilizzano l'accesso sequenziale a indici, è necessario che gli sviluppatori capiscano la struttura dei file ed utilizzino i file a indici per l'accesso ai dati.

La costruzione di un sistema di database utilizzando i file C-ISAM libera gli sviluppatori da quest'obbligo incorporando una struttura a file indici in un catalogo sotto forma di tabelle, colonne ed indici del catalogo.

## ➤ Tun SQL C-ISAM

C-ISAM si basa su funzioni C per interrogare ed aggiornare i file sequenziali. Il driver C-ISAM incluso in Tun SQL permette di vedere i file sequenziali come un database relazionale standard completo di tabelle, campi e chiavi. Diventa quindi possibile utilizzare istruzioni standard SQL per interrogare o aggiornare un database creato con file C-ISAM. Il driver C-ISAM traduce le istruzioni in funzioni C che seguono le operazioni necessarie sui file sequenziali.

Per passare da un vista dati sequenziale ad una vista dati relazionale, il driver C-ISAM aggiunge ai file dati ed indici standard C-ISAM alcuni file di database descrittivi conosciuti come **catalogo**. Questi sono file tipo C-ISAM (file dati **.dat** e file indici **.idx**): SysTables, SysColumns, SysIndexes e SysDefaults.



**sqltools** è un applicativo UNIX per creare e gestire database costruiti con C-ISAM. Funziona con lo stesso principio: le istruzioni SQL sono utilizzate per costruire il database e vengono tradotte in funzioni C prima di essere lette dal file system C-ISAM.



## ➤ Installazione del driver C-ISAM

Per installare **Tun SQL C-ISAM**, fare riferimento alla “**Guida di installazione Tun**”.

## Utilizzo di sqltools

### ➤ Utilizzo di sqltools

É possibile utilizzare **sqltools** “on-line” oppure da un’interfaccia semigrafica (finestre, menu).

Connettersi al server UNIX che contiene i file C-ISAM. Si raccomanda di creare un ID utente dedicato all’accesso ai file C-ISAM.

Spostarsi alla directory di installazione di **sqltools** ed eseguire l’applicazione digitando il comando:

```
sqltools  
per eseguire l’applicazione “on-line”,  
oppure  
sqltools -v  
per utilizzare l’interfaccia grafica
```

### ➤ Creazione del database

Il primo passo consiste creare il database contenente i dati dai file C-ISAM. Per fare questo utilizzare uno dei seguenti metodi:

- Scegliere **Database→Create** e digitare il nome del database che si desidera creare.
- Digitare il seguente comando nel riquadro sottostante (Finestra di immissione):

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

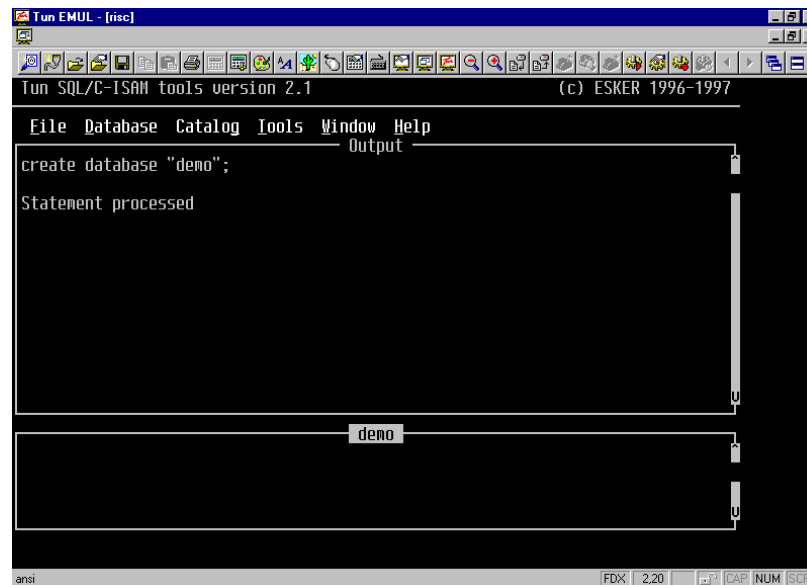
Questo comando crea una directory con il nome del database e l'estensione **.ism** nella directory indicata dalla variabile ISAM-PATH. Per ulteriori informazioni sulla variabile ISAM-PATH fare riferimento a **“Installazione del driver C-ISAM”** nella **“Guida all’installazione di Tun”**.

*Esempio:*

**Create database "demo";**  
*crea la directory demo.ism nella directory /TunSql/bases se ISAM-PATH=/TunSql/bases.*

Questa directory contiene i file C-ISAM .dat e .idx che descrivono il database: SysTables, SysColumns, SysIndexes e SysDefaults, per un totale di 4 file logici C-ISAM e 8 file del sistema operativo.

La finestra inferiore (Finestra di immissione) prende il nome del database:





**Nota:**

É possibile eseguire comandi di shell direttamente da **sqltools**.  
Immettere il comando preceduto da un punto esclamativo e seguito da un punto e virgola nel riquadro inferiore (Finestra di immissione).

*Esempio:*

```
!ls -a ;
```

Per immettere stringhe di caratteri in maiuscolo utilizzare i doppi apici.

*Esempio:*

```
!ls "/TunSql/locisam";
```

### ➤ Connessione ad un database esistente

Se si utilizza un database esistente, è possibile effettuare le operazioni sulle tabelle dopo la connessione.

Per fare questo, scegliere **Database→Connect** ed immettere il nome del database. Il riquadro inferiore (Finestra di immissione) prende il nome del database al quale si è connessi.

### ➤ Creazione di tabelle

Una volta creato il database (la directory **.ism** contiene i file descrittivi delle tabelle), è possibile procedere alla creazione di tabelle.

É possibile eseguire questa operazione utilizzando un paio di file C-ISAM esistenti (il file dati **.dat** ed il file indice **.idx**), oppure crearne di nuovi. Il comando utilizzato in questo secondo caso è differente, come lo sono le precauzioni da prendere: utilizzare il comando **create table** per creare i due file C-ISAM contemporaneamente alla tabella, ed il comando **define table** per creare una tabella da due file C-ISAM esistenti.

#### Creazione di tabelle e dei file C-ISAM

Per creare i due file C-ISAM contemporaneamente alla tabella, immettere nel riquadro inferiore (Finestra input), che ha ora come titolo il nome del database, il seguente comando:

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

Questo comando crea una tabella con nome "tablename" nel database. La tabella contiene i campi field1, field2, etc. con tipi type1, type2, etc. Fare riferimento alla lista dei tipi.

I due file C-ISAM dati e indice vengono creati nella directory del database (**database.ism**). I nomi di questi file utilizzano i primi sette caratteri del nome tabella ed un identificativo numerico unico attribuito automaticamente. L'estensione **.dat** viene aggiunta per i file dati e **.idx** per i file indici. Se il nome tabella è inferiore ai sette caratteri, questi nomi di file vengono completati con un segno di sottolineato ("\_").

*Esempio:*

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

*crea i file **table1\_100.dat** e **table1\_100.idx**. La tabella avrà il campo **field1** contenente un dato tipo **longint**, il campo **field2** contenente un dato di lunghezza massima di 25 caratteri, ed il campo **filler** contenente un dato di lunghezza massima 30 caratteri. La chiave primaria per questa tabella è **field1**.*

### Creazione di tabelle da un paio di file C-ISAM esistente

Per creare una tabella in un database di cui esistono già i file C-ISAM **.dat** e **.idx** immettere nel riquadro inferiore (Finestra di immissione), che ha ora come titolo il nome del database, il seguente comando:

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

Il comando crea una tabella con nome "tablename" dai file esistenti **filename.dat** e **filename.idx**.

#### **Nota importante:**

Prima di utilizzare la tabella (per esempio, prima di utilizzare il comando **select** sulla tabella), è necessario copiare i file specificati dalla chiave **file is** nella directory del database.

Il comando **define**, comunque, può essere eseguito anche se i file non sono stati copiati in questo directory. Si raccomanda inoltre di **non** copiare i file fino a che non viene eseguito il comando **create index**, se si desidera creare un indice per la tabella.



## ➤ Creazione di un indice

Per creare un indice per una e fino a otto colonne in una tabella, digitare il seguente comando nel riquadro inferiore (Finestra di immissione), che ha ora come titolo il nome del database, il seguente comando:

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

Questo comando crea l'indice "indexname" per le colonne **field1** e **field3** nella tabella "tablename".

## ➤ Strutture C

Ogni comando inviato a **sqltools** è tradotto in codice C per il file system C-ISAM. Per visualizzare la struttura C per una tabella, scegliere **Catalog**→**GetCStruct**.

Per esempio, il comando che crea **table1** produce la seguente struttura C:

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

In questo esempio, c'è un campo riservato (un array di **unsigned char**) lungo un byte. Questo campo è specifico per il sistema di gestione C-ISAM. Quando viene creata una tabella con il comando **create table**, **sqltools** aggiunge automaticamente questo campo riservato alla tabella.

## ➤ Validazione di una tabella creata da file C-ISAM esistenti

Quando viene creata una tabella da un paio di file C-ISAM esistenti (con il comando **define**), è necessario assicurarsi che la struttura della tabella creata sia conforme a quella dei file C-ISAM.

Per esempio, si è creata una tabella "table2" basata su file C-ISAM **filename.dat** e **filename.idx**, scritta in C. Questi file hanno la seguente struttura di record:

- Un campo composto da una variabile longint,

- Un campo composto da un array di 25 variabili carattere,
- Un campo composto da un array 30 variabili carattere.

Se si definisce la tabella "table2" come segue:

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

viene creata una tabella i cui record hanno la struttura:

- Un campo di tipo longint,
- Due campi array carattere di lunghezza 25,

che non corrispondono alla struttura generata dai file C-ISAM su cui è basata la tabella.

In questo caso, c'è una discordanza tra la tabella creata ed i file C-ISAM su cui è basata.

Per verificare che una struttura dati di una tabella corrisponda ai file C-ISAM originali, selezionare **Catalogo** → **Check Define**. Per verificare tutte le tabelle nel database, selezionare **Tools** → **Check Catalog**.

Nel nostro esempio, la finestra di output mostra i seguenti risultati:

```
Tun SQL/C-ISAM tools version 2.1 (c) ESKER 1996-1997
File Database Catalog Tools Window Help
Output
define table table2 file is filename (field1 longint, field2 char(25), filler
Statement processed
SQLCheckDefine ,, 'table2';
table_qualifier table_owner table_name table_is_ok remark
-----
null , "root", "table2" , 0 , "
Table size different from file record size (55 < 60)"
1 column(s) retrieved
```



Viene visualizzato il seguente messaggio:

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

Il messaggio avverte che la tabella **table2** è stata definita diversamente dai file C-ISAM "filename" su cui è basata. Il comando **define** dovrebbe essere come segue:

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

### ➤ Visualizzazione della struttura C di una tabella

È possibile visualizzare la struttura C di una tabella, cioè le diverse colonne create (nomi, tipi dati e lunghezza), così come le informazioni riguardanti la gestione dei dati (ad esempio le chiavi primarie).

Per fare questo, selezionare **Catalog**➔**GetCStruct** e digitare il nome della tabella di cui si desidera visualizzare la struttura C. Per visualizzare la struttura C di tutte le tabelle nel database, selezionare **Tools**➔**List Structures**.

L'esempio che segue mostra i risultati di questo comando per una tabella creata con una chiave primaria:

Il seguente comando crea la tabella:

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

La struttura C generata è:

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

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

Il primo pezzo di codice mostra la struttura della tabella (i campi), mentre il secondo mostra le chiavi primarie con le varie assegnazioni.

## ➤ Informazioni sul catalogo

É possibile ottenere informazioni sul catalogo utilizzando il menu **Catalog**.

Il menu opzioni fornisce informazioni su:

- **TypeInfo**: Ogni tipo di dati è identificato da un numero, come definito dallo standard ODBC. Digitare il numero per il tipo di dati che si desidera verificare, oppure immettere 0 per visualizzare la lista dei tipi di dati.
- **Tables**: É possibile interrogare il catalogo per le informazioni sulle tabelle, utilizzando il nome dell'utente che ha creato la tabella, il nome stesso della tabella oppure il tipo tabella (tabella di sistema, sinonimo, ecc.). Immettere il carattere % per includere nell'interrogazione tutte le tabelle o i tipi di tabelle.
- **Columns**: É possibile interrogare il catalogo per informazioni sulle colonne, utilizzando il nome dell'utente che ha creato la tabella, il nome della tabella stessa, oppure il nome della colonna. Immettere il carattere % per includere nell'interrogazione tutte le tabelle o le colonne.
- **Statistics**: É possibile ottenere statistiche sui dati nel catalogo.



- **PrimaryKeys:** È possibile interrogare il catalogo per chiavi primarie di tabella, utilizzando il nome dell'utente che ha creato la tabella, il nome della tabella stessa. Immettere il carattere % per includere nell'interrogazione tutte le tabelle.

Per ulteriori informazioni sulle opzioni del menu **Catalog**, fare riferimento alla sezione sul catalogo di sistema nel manuale di riferimento dello standard ODBC.

### ➤ **Modifica di una tabella**

La lunghezza dei record in una tabella viene impostata alla sua creazione. Di conseguenza, non è possibile aggiungere o cancellare record se ciò coinvolge la lunghezza totale.

Se si desidera cambiare la struttura di una tabella, è necessario prima cancellare la tabella rispettando le precauzioni descritte in “**Cancellazione di una tabella**”.

### ➤ **Cancellazione di una tabella**

Ci sono due modi per cancellare una tabella dal catalogo:

È stata creata una tabella utilizzando il comando **create table**: utilizzare il comando **drop table** per cancellarla. **Nota:** Questo comando rimuove tutti i riferimenti alla tabella dai file di catalogo e cancella il paio di file C-ISAM collegati alla tabella.

È stata definita una tabella utilizzando il comando **define table**: utilizzare il comando **undefine table** per cancellare il comando **define**. Questa istruzione rimuove dal catalogo soltanto tutti i riferimenti alla tabella.

**Nota:**

È possibile utilizzare il comando **drop table** per una tabella definita da **define table**. Comunque, l'istruzione **drop table** cancella il paio di file C-ISAM specificati dalla parola chiave **file is** se questi si trovano nella directory del database. È quindi necessario fare attenzione nell'uso del comando **drop table** se la tabella è stata creata da file C-ISAM esistenti.

Se si desidera mantenere il paio di file C-ISAM associati ad una tabella cancellata utilizzando il comando **drop table**, è necessario prima copiare i file in una directory differente. In questo modo, dopo la cancellazione della tabella è possibile utilizzare queste copie di riserva.

### ➤ **Manutenzione dei file C-ISAM**

Quando vengono create delle tabelle da file C-ISAM esistenti (utilizzando **define table**), per utilizzare queste tabelle è necessario copiare questi file nella directory del database.

Comunque, se questi file vengono utilizzati ed aggiornati, in altre applicazioni, è utile poter utilizzare gli aggiornamenti nel proprio database. Per fare questo, bisogna creare collegamenti (link) simbolici dal database ai file C-ISAM esistenti from the database invece di farne copie.

*Esempio:*

*Viene creato il database **dbtest** nella directory **/TunSql**. Per le tabelle in questo database, vengono utilizzati i file **filename.dat** e **filename.idx**, che si trovano nella directory **/data** e sono utilizzati da altre applicazioni.*

*Vengono creati dei collegamenti simbolici a questi file nella directory **/TunSql/dbtest.ism** (la directory del database) con il seguente comando:*

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

### ➤ **Cancellazione di database**

Per cancellare un database, selezionare **Database→Drop** ed immettere il nome del database da rimuovere, oppure digitare nel riquadro inferiore (Finestra di immissione) il seguente comando:

```
drop database databasename
```

### ➤ **Salvataggio risultati**

È possibile salvare i risultati visualizzati nel riquadro superiore (Finestra di output) in un file di testo con estensione **.res**.





Per fare questo, selezionare **File**→**Save as...** e scegliere la posizione ed il nome del file da salvare.

### ➤ **Esecuzione di script**

È possibile utilizzare l'opzione **File**→**Execute** per eseguire uno script SQL a condizione che vengano utilizzati comandi SQL supportati da **sqltools**.



**PARTE 2**  
**RIDEFINIZIONE DEL**  
**DATABASE**



# REVAMPING

---

## Database virtuali

I sistemi RDBMS (Relational Database Management Systems, sistemi di gestione database relazionali) rappresentano la soluzione moderna più diffusa per la memorizzazione di dati in formato strutturato. I database RDBMS consentono di immagazzinare i dati delle società in modo tale da poterli eventualmente aggiornare tramite una semplice operazione. La massa di dati raccolti in questo modo è utile anche per il numero potenzialmente vasto di utenti che desiderino estrarre da tali basi di dati informazioni per le proprie attività (indicatori delle prestazioni, statistiche, sistemi esperti). Per l'aggiornamento e l'interrogazione dei database viene utilizzato il linguaggio SQL.

La struttura dei database, che costituiscono il cuore dei sistemi informativi, può tuttavia rendere difficile l'accesso alle informazioni perché:

- I database contengono un numero di tabelle e record troppo alto per l'utente finale medio, spesso interessato solo ad una parte dei dati.
- La struttura di un database è in ogni caso complessa cosicché all'utente si richiede una notevole esperienza per capirne il funzionamento.
- L'ambiente informatico che gestisce i database non è di tipo intuitivo (user-friendly). I nomi delle tabelle e dei record, ad esempio, vengono di rado indicati con nomi che ne indicano il tipo.
- La manipolazione e l'uso dei dati presuppongono una conoscenza del linguaggio SQL per interrogare i database e per estrarre i dati desiderati.

Per superare questi ostacoli e facilitare l'accesso ai database sono state realizzate diverse soluzioni (come l'integrazione di interfacce grafiche negli strumenti di interrogazione).

Il passo successivo nel senso della semplificazione, rendendo disponibili solo le informazioni di cui l'utente ha bisogno nella forma più adatta per il suo ambiente di lavoro, prevede che l'utente finale non debba più avere una competenza tecnica per accedere ai database.

Ciò ha prodotto i seguenti vantaggi:

- Migliore produttività: l'utente finale è più autonomo nell'uso dei dati, mentre l'analisi e il decision making richiedono meno tempo in quanto più semplici.
- Informazioni più pertinenti: utilizzando solo i dati che gli servono e che è in grado di gestire, l'utente affina la propria capacità di analisi e sintesi e ottiene risultati più interessanti

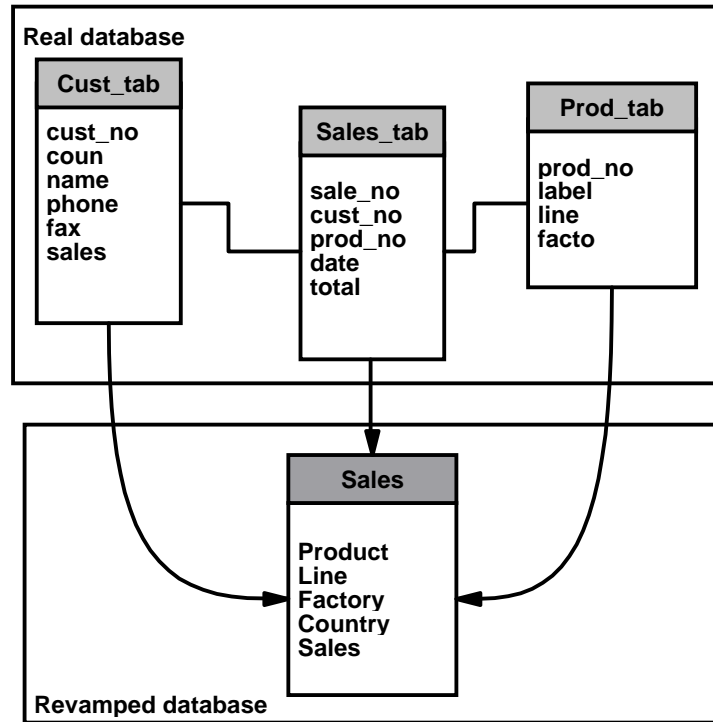
### ➤ **Ridefinizione**

La ridefinizione o "revamping" del database consiste nel costruire un database virtuale adattato all'ambiente dell'utente a partire dal database preesistente. Sebbene non esista come database reale, la nuova struttura viene vista dall'utente come un database normale in cui tuttavia le tabelle e i campi corrispondono esattamente alle sue esigenze: il database contiene in forma adeguata unicamente le informazioni necessarie all'analisi (nomi dei dati comprensibili, funzioni predefinite).

Il database ridefinito viene perfezionato dall'amministratore che riconfigura le tabelle e i campi partendo da un database reale.



Ad esempio:



Nell'esempio riportato sopra, il database reale contiene tre tabelle: "Cust tab" (tabella dei clienti), "Sales tab" (tabella delle vendite) e "Prod tab" (tabelle dei prodotti).

L'amministratore definisce una tabella virtuale presentando i dati delle vendite per prodotto, per linea di prodotti, per impianto produttivo e per paese.

La tabella virtuale "Vendite" contiene i campi seguenti:

- Prodotto (campo reale: "prod\_tab.label»)
- Linea (campo reale: "prod\_tab.line")
- Fabbrica (campo reale: "prod\_tab.fact")
- Paese (campo reale: "cust\_tab.coun")
- Vendite (campo reale: "sales\_tab.total")

La tabella virtuale crea un join tra le tabelle "Prod\_tab" e "Sales\_tab" utilizzando il campo comune "prod\_no", e un join tra le tabelle "Sales\_tab" e "Cust\_tab" utilizzando il campo comune "cust\_no".

## Ridefinizione in Tun SQL

**Tun SQL** è in grado di eseguire l'amministrazione e l'uso di database virtuali grazie ai due componenti seguenti:

- L'applicazione **Tun DB Revamp** dell'amministratore per la ridefinizione del database.
- Il driver ODBC virtuale per l'accesso ai database ridefiniti dall'amministratore.

### ➤ Applicazione Tun DB Revamp dell'amministratore

Lo scopo del database virtuale **Tun SQL** è quello di consentire all'utente finale di ridefinire le informazioni in modo da contestualizzarle per un particolare "ambiente" di lavoro.

Grazie all'interfaccia grafica di uso intuitivo l'amministratore può ridefinire il numero di "ambienti" desiderati per più utenti o tipi di utente. Ogni ambiente si basa sulle mansioni di lavoro: è cioè possibile fare in modo che gli addetti alla contabilità vedano solo le tabelle relative alla contabilità, i venditori le tabelle sulle vendite, ecc.

Ciascun ambiente può essere visualizzato dall'interfaccia ODBC che trasforma le query in interrogazioni specifiche (si veda il diagramma relativo a questa architettura nella sezione "**Driver ODBC virtuale**").

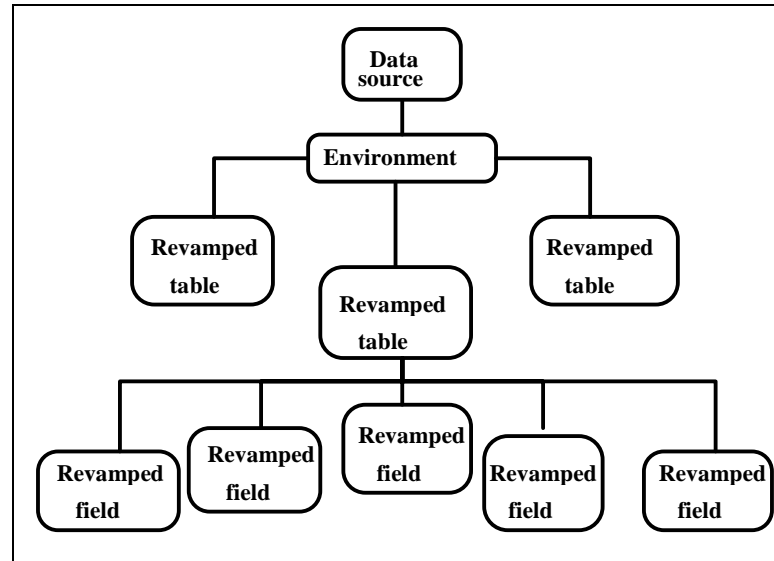
Il modello del database virtuale è strutturato nel modo seguente:

- Uno o più ambienti definiti partendo da un'origine dati reale e contenenti una selezione di tabelle del database reale secondo le esigenze dell'utente.
- Le tabelle definite all'interno di un ambiente sono native di un database reale oppure il risultato di operazioni di "join" tra due o più tabelle (concetto di "vista").
- Ciascuna tabella contiene esclusivamente i campi richiesti dall'utente.





- I campi sottoposti a ridefinizione sono campi preesistenti appartenenti a tabelle reali o campi ricalcolati che semplificano ulteriormente l'uso del database.
- Alle tabelle e ai campi ridefiniti possono essere assegnati nomi comprensibili all'utente finale (ad esempio "Cust\_tab" può essere chiamato "Tabella clienti" e "Cust\_no" "Numero cliente").



Le tabelle ridefinite in un ambiente non esistono fisicamente all'interno del database. Tuttavia il database ridefinito è memorizzato con un sistema di indicizzazione all'interno di tre tabelle supplementari create nel database:

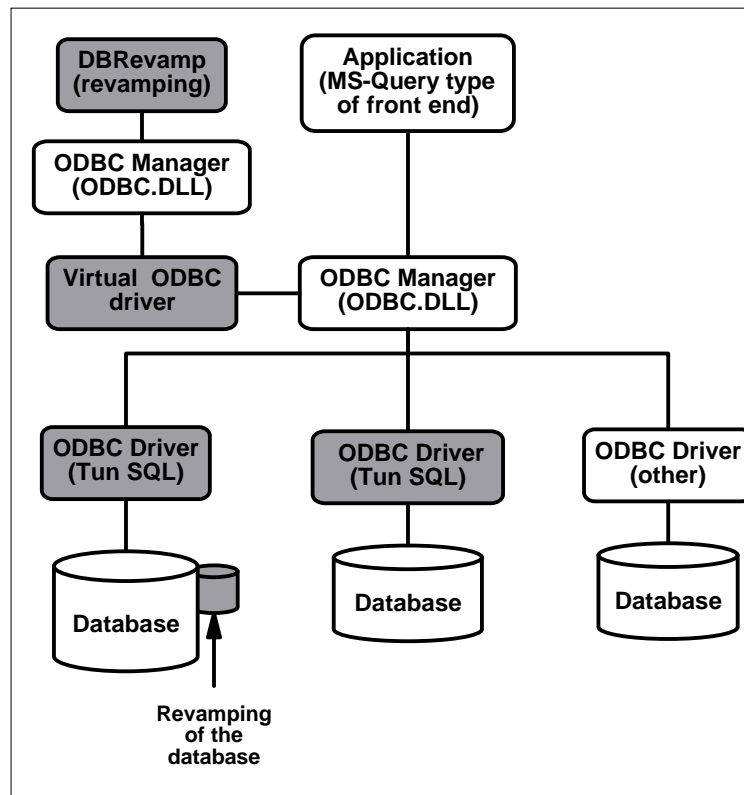
- La tabella d'ambiente contenente l'elenco degli ambienti con l'indicazione di "nome ambiente" e "descrizione".
- Una tabella contenente l'elenco delle tabelle ridefinite, ciascuna indicizzata per nome, descrizione e nome dell'ambiente di appartenenza.
- Una tabella dei campi ridefiniti, ciascuno indicizzato per nome, descrizione, origine (campo preesistente, dati elaborati, concatenazione dei dati) e nome della tabella virtuale di appartenenza.

Un database che sia stato ridefinito contiene sempre queste tre tabelle aggiuntive.

## ➤ Driver ODBC virtuale

Per l'utente finale l'interrogazione di un database virtuale avviene in modo analogo all'interrogazione di un database reale in modalità di sola lettura. Tale trasparenza è dovuta all'integrazione in **Tun SQL** di uno speciale driver ODBC per database virtuali.

Quando il manager ODBC (ODBC.DLL) riceve interrogazioni da un ambiente particolare, le trasmette al driver ODBC virtuale che le converte in interrogazioni appropriate per il database reale. Successivamente, il driver ODBC virtuale ritorna le interrogazioni così convertite al manager ODBC, che a sua volta le reindirizza al driver ODBC standard del database reale.



## USO DI TUN DB REVAMP

---

Il presente capitolo fornisce una descrizione dei principali comandi del programma **Tun DB Revamp**.

### Opzioni generali

#### ➤ Scelta della lingua

Per selezionare la lingua utilizzata nell'interfaccia del programma selezionare il comando ?→**Lingua...** e la lingua desiderata.

#### ➤ Opzioni di visualizzazione



Per cambiare i controlli visualizzati nella finestra principale del programma **Tun DB Revamp**:

- Selezionare o annullare il comando **Visualizza→Barra degli strumenti** del menu principale per visualizzare/nascondere la barra strumenti.
- Selezionare o annullare il comando **Visualizza→Barra di stato** del menu principale per visualizzare/nascondere la barra di stato.
- Selezionare o annullare il comando **Visualizza→Barra delle proprietà** del menu principale per visualizzare/nascondere la barra delle proprietà degli oggetti.

#### ➤ Copia di un oggetto


Per copiare un oggetto utilizzare uno dei seguenti metodi:

1. Metodo "**drag and drop**": selezionare l'oggetto da copiare e trascinare il mouse nel punto in cui copiarlo tenendo premuto il tasto.

2. Con i comandi del menu principale **Modifica→Copia** (per copiarlo) e **Modifica→Incolla** (per incollarlo nel punto desiderato).
3. Con i comandi **Copia** e **Incolla** del menu contestuale (visualizzabile con il tasto destro del mouse).
4. Con le combinazioni di tasti **Ctrl-C** (copia) e **Ctrl-V** (incolla).
5. Con i pulsanti della barra strumenti  (copia)  (incolla).

### ➤ Eliminazione di un oggetto

Per eliminare un oggetto, selezionarlo e:

1. Utilizzare il comando del menu principale **Modifica→Cancella**.
2. Utilizzare il comando del menu contestuale **Cancella**.
3. Utilizzare i tasti **Canc**.
4. Fare clic sul pulsante  della barra strumenti.

### ➤ Modifica del nome di un oggetto


Per modificare il nome di un oggetto, selezionarlo e utilizzare uno dei metodi seguenti:

1. Utilizzare la scheda **Generico** della barra delle proprietà.
2. Utilizzare il tasto funzione **F2** della tastiera e sostituire il nome precedente con quello nuovo.
3. Fare clic di nuovo sull'oggetto e procedere come al punto 2.

### ➤ Memorizzazione delle modifiche

Per memorizzare le modifiche apportate alle proprietà, premere Invio quando il cursore si trova nella finestra di dialogo relativa oppure premere il pulsante **Applica**.

### ➤ Uso della Guida

Per accedere alla guida in linea o per ottenere ulteriori informazioni su DBRevamp, fare clic sul comando del menu principale **?→A proposito di DBRevamp** oppure utilizzare il pulsante della barra strumenti .

### ➤ Uscita da Tun DB Revamp

Per uscire dall'applicazione, fare clic sul comando **File→Esci**.

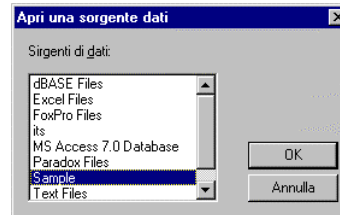


## Importazione di ambienti di sorgenti di dati

Per ridefinire un database reale è necessario selezionare la sorgente di dati corrispondente. A tal fine utilizzare il comando **File→Importa...**

oppure fare clic sul pulsante  della barra degli strumenti.

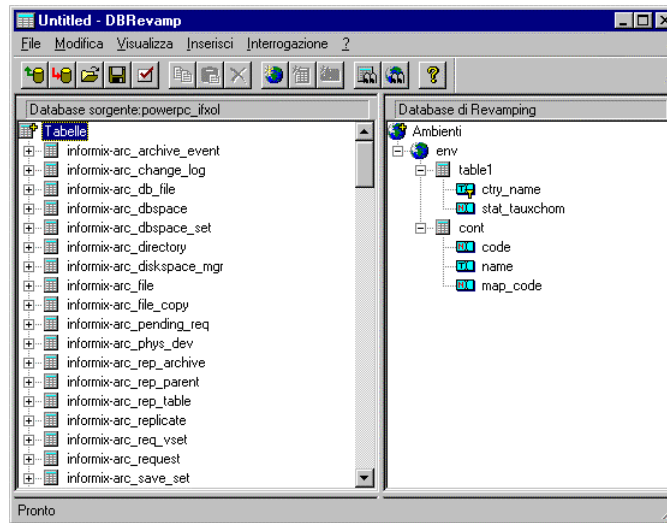
Appare la seguente finestra di dialogo:



La finestra mostra la lista delle sorgenti di dati dichiarate su PC. Per creare una sorgente di dati, vedere "**Creazione di sorgenti di dati**". Siccome le sorgenti dati virtuali (ottenute tramite la ridefinizione di un database reale) non possono essere ridefinite, esse non compaiono nell'elenco. Al contrario, esse compaiono nell'elenco delle sorgenti dati disponibili all'utente finale nelle applicazioni Windows che le utilizzano (ad esempio Microsoft Query).

Selezionare la sorgente di dati che si desidera utilizzare.

Verrà visualizzata una finestra **Tun DB Revamp** simile alla seguente.




Le tabelle del database reale compaiono nella parte sinistra della finestra.

Se il database in questione non è stato ridefinito con **Tun DB Revamp**, la parte destra della finestra conterrà un ambiente vuoto, "Nuovo ambiente", da configurare.


Se viceversa il database è già stato ridefinito con **Tun DB Revamp**, nel qual caso si tratta di un aggiornamento del database virtuale, la parte destra della finestra conterrà l'elenco degli ambienti già creati e il relativo contenuto.

## Creazione di un ambiente



Per definire un nuovo ambiente per la sorgente di dati desiderata, selezionare il percorso base degli ambienti (chiamato **Ambienti**) e scegliere **Inserisci** → **Nuovo Ambiente** dal menu principale. È anche possibile fare clic sul pulsante della barra strumenti .




Per definire un nuovo ambiente per la sorgente di dati desiderata, selezionare il percorso base degli ambienti (chiamato **Ambienti**) e scegliere **Inserisci**→**Nuovo Ambiente** dal menu principale. È anche possibile fare clic sul pulsante della barra strumenti .

Digitare un nome e (facoltativamente) una descrizione dell'ambiente.

## Creazione di una tabella virtuale



Per creare una tabella virtuale in un ambiente, selezionare l'ambiente ed effettuare quanto segue:

- Scegliere **Inserisci**→**Nuova Tabella** dal menu principale oppure selezionare **Nuova Tabella** dal menu di contesto dell'ambiente. È anche possibile fare clic sul pulsante della barra strumenti .
- Scegliere **Visualizza**→**Proprietà** per accedere alla finestra delle proprietà del campo appena creato (se non è già visualizzata).
- Nella sezione **Generale** nella **Finestra Proprietà**, immettere un nome ed una descrizione eventuale per la tabella. È inoltre possibile utilizzare il tasto funzione F2 per rinominare una tabella selezionata.

Se si desidera che la tabella virtuale contenga una tabella reale, in parte o per intero, è possibile copiare la tabella reale nell'ambiente scelto; in questo modo verranno copiati anche tutti i campi della tabella virtuale. Per fare questo:

- Utilizzare uno dei metodi descritti nell'introduzione (**drag 'n drop**, **Copia/Incolla**, tasti di scelta rapida e pulsanti della barra strumenti) per selezionare la tabella reale nel database di origine e copiarla nell'ambiente di destinazione.
- Eliminare i campi non richiesti nel database virtuale oppure modificarli secondo quanto descritto nella sezione "**Creazione di un campo**".
- È inoltre possibile modificare i nomi degli oggetti copiati (tabelle e campi) e allegare a ciascuno una descrizione nella corrispondente scheda **Generico**.

## Creazione di un campo



In una tabella virtuale, è possibile:

- Inserire un campo esistente da un database reale, senza cambiare la sua definizione.
- Creare un nuovo campo virtuale da campi reali del database.

In una tabella virtuale, è possibile:

- Inserire un campo esistente da un database reale, senza cambiare la sua definizione.
- Creare un nuovo campo virtuale da campi reali del database.


### ➤ Campi preesistenti

È possibile copiare un campo preesistente da una tabella di un database reale direttamente nella tabella virtuale. Per fare questo:

- Utilizzare uno dei metodi descritti nell'introduzione (drag 'n drop, Copia/Incolla, tasti di scelta rapida o pulsante della barra strumenti) per selezionare il campo nella tabella del database reale e collocarlo nella tabella virtuale del database ridefinito.
- È possibile cambiare il nome del campo e assegnargli una descrizione nella relativa scheda **Generico** (oppure utilizzando il tasto funzione **F2**).

### ➤ Nuovo campo

Per definire un nuovo campo in una tabella virtuale, selezionare la tabella virtuale ed eseguire quanto segue:

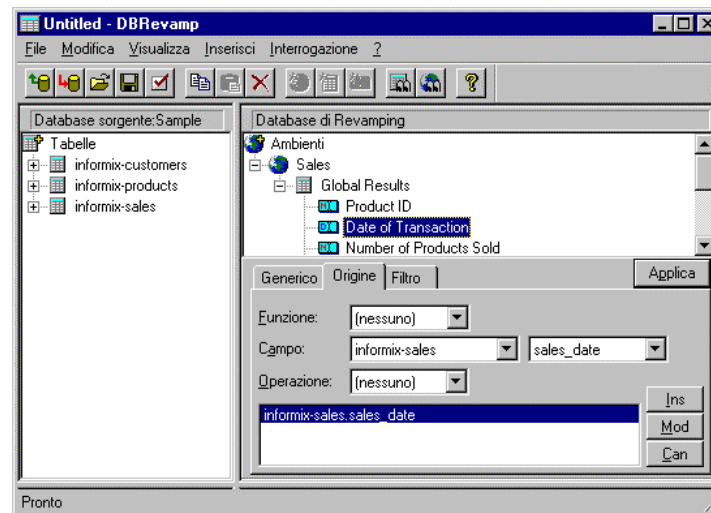
- Scegliere **Inserisci**→**Nuovo Campo** dal menu principale oppure selezionare **Nuovo Campo** dal menu di contesto della tabella. È anche possibile fare clic sul pulsante della barra strumenti .
- Scegliere **Visualizza**→**Proprietà** per accedere alla finestra delle proprietà del campo appena creato (se non è già visualizzata).
- Digitare un nome e (facoltativamente) una descrizione nella scheda **Generico**.





Fare clic sulla scheda **Origine**. A questo punto è possibile:

- Aggiungere al campo una funzione: selezionare la funzione scelta nella casella di riepilogo **Funzione**. Le funzioni disponibili sono: **Somma**, **Minimo**, **Massimo**, **Numero**, **Media** o **nessuno**.
  - Aggiungere al campo nuovo un valore da un campo esistente del database reale oppure la funzione selezionata sopra; selezionare la tabella e il campo reali desiderati nelle due caselle di riepilogo **Campo**.
  - Aggiungere un'operazione al campo selezionato sopra; scegliere l'operatore desiderato dalla casella di riepilogo **Operazione**. Le operazioni disponibili sono: +, -, \*, / o nessuno. Per concatenare i caratteri utilizzare l'operatore +.
- Per aggiungere queste opzioni alla definizione del campo fare clic sul pulsante **Ins**.



*Esempio 1:*

*Si supponga di avere a disposizione la tabella reale "res\_tab" contenente quattro campi (res1, res2, res3 e res4) corrispondenti ai risultati trimestrali di un anno in particolare e di voler definire il campo "Risultato" in una tabella del database virtuale contenente la somma dei quattro campi dell'anno.*

Nella scheda **Origine**, per il campo "Risultato":

- Selezionare la tabella "res\_tab" e il campo "res1" nelle caselle di riepilogo **Campo**.
- Selezionare l'operatore + nella casella di riepilogo dell'opzione **Operazione**.
- Fare clic sul pulsante **Mod** per sostituire i valori predefiniti. Il nuovo valore è "res\_tab.res1 +".
- Nelle caselle di riepilogo dell'opzione **Campo** selezionare la tabella "res\_tab" e il campo "res2".
- Nella casella di riepilogo **Operazione** selezionare nuovamente l'operatore +.
- Per aggiungere la voce "res\_tab.res2 +" appena creata fare clic sul pulsante **Ins**.
- Ripetere la stessa operazione per il campo "res3".
- Per il campo "res4", selezionare l'operatore **Nessuno** invece di +.
- Il campo "Risultato" viene definito in base all'elenco seguente:

res\_tab.res1+  
res\_tab.res2+  
res\_tab.res3+  
res\_tab.res4

Il campo "Risultato" contiene perciò la somma dei quattro campi "res1", "res2", "res3" e "res4".

*Esempio 2:*

Si desidera il campo Risultato che mostri la somma delle vendite annuali. Per averlo, trovare la somma di tutti i campi res1, res2,... quindi trovare la somma di questi per avere il risultato desiderato.

Nella sezione **Origine** del campo Risultato:

- Selezionare la funzione **Somma** dalla casella **Funzione** del campo.
- Selezionare la tabella res\_table ed il campo res1 da ognuna delle caselle **Opzione** del campo.
- Selezionare l'operatore + dalla casella **Operazione**.
- Fare clic sul pulsante **Mod** per sostituire i valori predefiniti. Il nuovo valore è "res\_tab.res1 +".
- Ripetere l'operazione per i campi "res2" e "res3". Per il campo "res4", selezionare l'operatore "nessuno" invece dell'operatore "+" della lista **Operazione**.



Per modificare un elemento nella definizione di un campo utilizzare il pulsante **Mod**; in questo caso l'elemento evidenziato viene sostituito dai valori selezionati sopra. Fare clic sul pulsante **Can** per eliminare l'elemento evidenziato.

Dopo aver definito un campo, fare clic sul pulsante **Applica** per rendere attive le nuove opzioni.

Subito dopo aver definito un nuovo campo virtuale, ricordarsi di definire i join tra la/le tabella/e utilizzate per creare la tabella virtuale (se disponibili). Consultare al riguardo la sezione "**Collegamenti tra tabelle**".

Per controllare che il calcolo assegnato al campo creato sia quello effettivamente desiderato utilizzare la funzione di interrogazione di **Tun DB Revamp**. Vedere la sezione "**Interrogazione di database reali e virtuali**".

## Assegnazione ai campi di filtri

La definizione di un campo virtuale può essere completata con un filtro; ciò significa che è possibile definire una condizione per il calcolo del valore del campo. Il filtro corrisponde alle limitazioni imposte ad una interrogazione (come in **MS Query**).

*Esempio:*

*Supponiamo di voler calcolare la somma dei campi "res1" quando il campo "res2" è maggiore di uno specifico valore. La condizione posta su "res2" costituisce un filtro.*

**Tun DB Revamp** consente di attribuire un filtro ai campi virtuali da utilizzare quando l'utente finale usa il campo. Un filtro può essere:

- Statico: il valore del filtro è fisso.
- Dinamico: gli utenti specificano i valori desiderati quando sottopongono la interrogazione.

Per assegnare un filtro ad un campo virtuale selezionare il campo e fare clic sulla scheda **Filtro** nella **Property Box**. Procedere quindi nel modo seguente:

- Digitare l'etichetta per il filtro nel campo **Etichetta**. Per i campi statici l'etichetta è facoltativa. Per i campi dinamici, l'etichetta deve indicare la funzione del filtro per la quale l'utente deve indicare un valore.
- Selezionare la tabella e il campo ai quali applicare il filtro dalle caselle di riepilogo **Campo**.
- Selezionare l'operatore di confronto dalla casella di riepilogo a discesa **Comp**.
- Per i filtri statici specificare il valore del filtro nel campo **Valori**. Per i filtri dinamici inserire un punto di domanda (?).
- Fare clic su **Ins** per inserire il criterio scelto.

È possibile creare un set di condizioni o criteri; i criteri vanno definiti come descritto in precedenza. Selezionare **And** o **Or** per aggiungere i criteri.

Per controllare che il filtro assegnato al campo creato sia quello effettivamente desiderato utilizzare la funzione di interrogazione di **Tun DB Revamp**. Vedere la sezione "**Interrogazione di database reali e virtuali**".

Se il filtro è di tipo dinamico le interrogazioni al campo virtuale comportano la visualizzazione di una finestra simile alla seguente:

The screenshot shows a dialog box with a blue title bar containing the text "Per favore immettere i seguenti parametri". The main area of the dialog is light gray and contains two rows of input fields. Each row starts with a label: the first is "label" and the second is "E 2". To the right of each label is a small dropdown menu containing an equals sign (=). Further right is a white text input box. To the right of each text box is a button labeled "Valori...". At the bottom of the dialog, there are two buttons: "Annulla" and "OK".

Specificare il valore richiesto per applicare il filtro al campo virtuale. Fare clic sul pulsante **Valori...** per visualizzare l'elenco dei valori possibili per il campo in oggetto.



## Collegamenti tra tabelle

I campi definiti in una tabella virtuale sono desunti da una o più tabelle del database reale.

Per ogni tabella virtuale è essenziale definire i collegamenti tra le tabelle reali dalle quali provengono i campi contenuti in essa. Tale definizione consente di creare i join tra le tabelle reali quando l'utente interroga il database virtuale. I collegamenti possono essere diretti o indiretti, cioè tra due tabelle o tra più tabelle fra di loro.

### ➤ Definizione dei collegamenti

Il modo più semplice per definire i collegamenti è contestualmente alla definizione dei campi della tabella virtuale. Tutte le tabelle reali utilizzate per la definizione dei campi devono essere collegate direttamente o indirettamente alle altre tabelle reali utilizzate dalla tabella virtuale.

Per definire collegamenti tra tabelle reali per la stessa tabella virtuale, selezionare la tabella virtuale ed eseguire quanto segue:

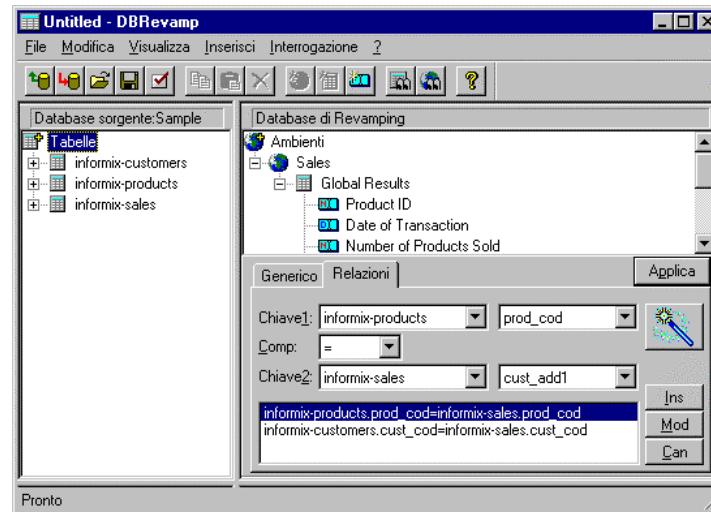
- Fare clic sulla sezione **Relazioni** della tabella virtuale.
- Per ogni tabella reale selezionare la tabella reale ed il campo da utilizzare quale collegamento con l'altra tabella utilizzando le liste nelle caselle **Chiave1** per la prima tabella reale, e **Chiave2** per la seconda tabella reale.

**Nota:**

I nomi dei due campi di collegamento tra le tabelle possono essere diversi anche se indicano gli stessi dati.

- Selezionare un operatore di confronto nella casella di riepilogo **Comp.**

- Per aggiungere il collegamento all'elenco dei collegamenti della tabella virtuale fare clic sul pulsante **Ins**.




Dopo aver cambiato i valori dell'elemento selezionato è possibile modificare un collegamento con il pulsante **Mod**. Per eliminare l'elemento evidenziato fare clic sul pulsante **Canc**.

Per convalidare l'elenco di collegamenti fin qui definiti fare clic sul pulsante **Applica**.

### ➤ Verifica dei collegamenti

Il programma **Tun DB Revamp** include una funzione per verificare i collegamenti definiti dall'amministratore tra le tabelle reali utilizzate per realizzare la tabella virtuale.

Per ogni tabella virtuale è possibile verificare in modo semplice se le tabelle reali utilizzate sono collegate e se i collegamenti definiti costituiscono un insieme coerente.

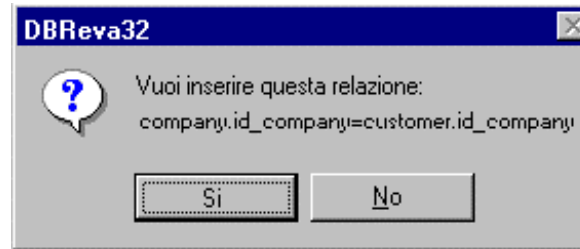
Per eseguire la verifica fare clic sul pulsante  della scheda **Relazioni**.



Il programma **Tun DB Revamp** provvederà ad esaminare tutti i collegamenti definiti dall'amministratore e a rilevare l'eventuale mancanza di collegamenti diretti o indiretti che isolano dal resto una o più tabelle specifiche.

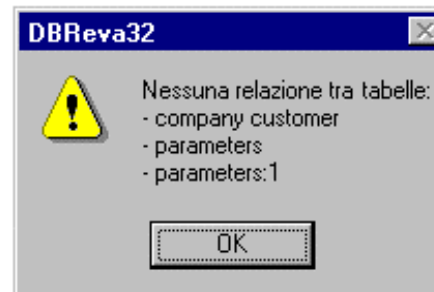
Se risulta mancante un collegamento tra due tabelle, **Tun DB Revamp** tenta di collegarle utilizzando due campi con lo stesso nome.

Se i due campi esistono, **Tun DB Revamp** propone di definire il collegamento nel modo seguente:



Nella maggior parte dei casi il collegamento proposto sarà adatto. Tuttavia se si ritiene che i due campi proposti da DBRevamp non debbano collegare le tabelle, definire il collegamento manualmente nel modo descritto nella sezione "**Definizione dei collegamenti**".

D'altro canto, se due tabelle non collegate non hanno campi con lo stesso nome **Tun DB Revamp** visualizza un elenco delle tabelle non collegate:




In questo caso definire il collegamento in modo manuale come descritto nella sezione "**Definizione dei collegamenti**".

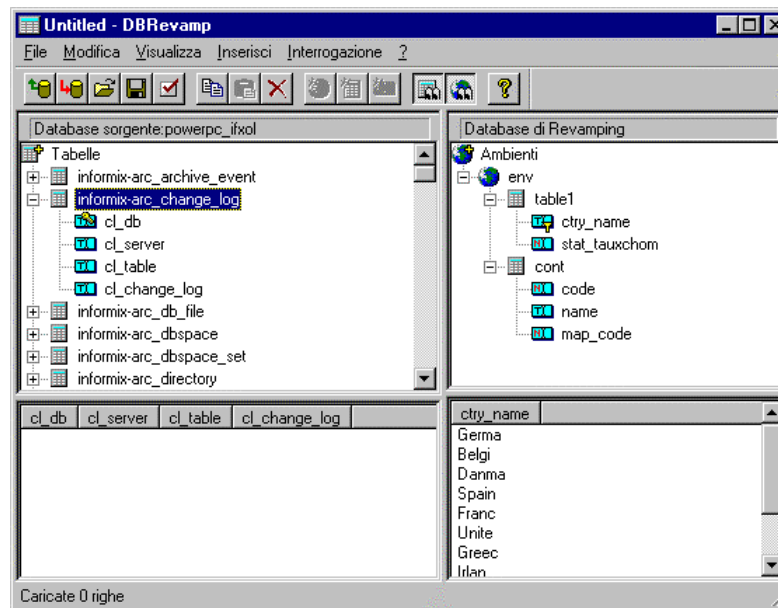
## Interrogazione di database reali e virtuali

**Tun DB Revamp** comprende una funzione di interrogazione per tabelle e campi di database reali e virtuali.

Tale funzione può essere utilizzata per visualizzare un campo o una tabella di un database reale o virtuale direttamente da **Tun DB Revamp** senza usare uno strumento di interrogazione come ad esempio **MS Query**.

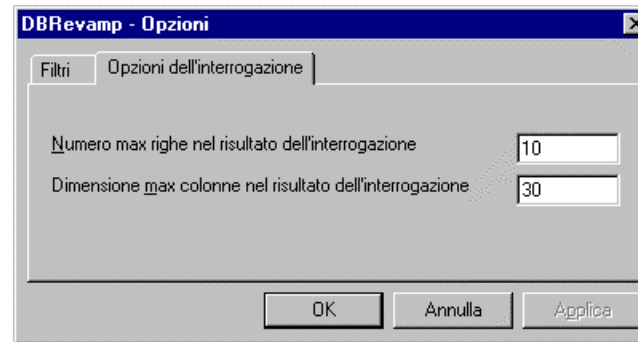
Per interrogare una tabella (o un campo) nel database reale scegliere il comando **Interrogazione→Source** (o **Interrogazione→Environment**) dal menu principale oppure fare clic sulla barra degli strumenti sul pulsante .

Si aprirà un riquadro sotto il riquadro del database reale o virtuale, a seconda del comando scelto. È possibile scegliere contemporaneamente i due comandi.





Per limitare il numero di record e la larghezza delle colonne durante l'interrogazione di una tabella o di un campo scegliere il comando **Visualizza**→**Opzioni...** dal menu principale e fare clic sulla scheda **Opzioni dell'interrogazione**:



Indicare il numero massimo di record da visualizzare nel campo **Numero max righe nel risultato dell'interrogazione**.

Indicare la larghezza di colonna massima visualizzabile nel campo **Dimensione max colonne nel risultato dell'interrogazione**.

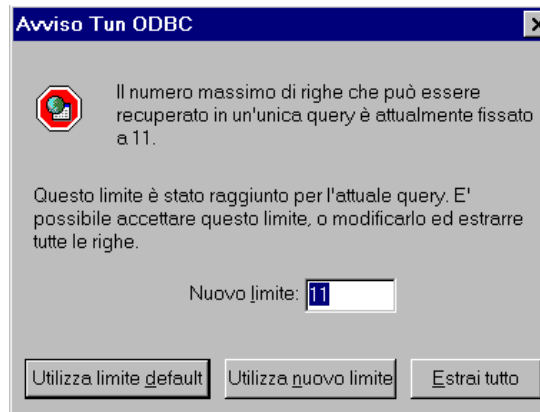
Nota:

Se al momento della creazione della sorgente dei dati sono stati definiti dei limiti (vedere la sezione "Creazione di una sorgente di dati"), i limiti avranno la precedenza rispetto a quanto indicato in Opzioni dell'interrogazione.

*Esempio*

*Supponiamo che durante la configurazione della sorgente reale dei dati sia stato impostato un limite variabile di 11-15 linee.*


Se si interroga un database reale contenente più di 11 record verrà visualizzato un avviso simile al seguente:



L'avviso è determinato dal tipo di limite utilizzato.

## Convalida di un ambiente

Il programma **Tun DB Revamp** comprende una funzione per verificare la consistenza tra il contenuto degli ambienti creati dall'amministratore e il contenuto del database reale utilizzato. Tale funzione è particolarmente utile dopo modifiche alla struttura del database reale (ad esempio dopo la modifica o eliminazione di un campo) non integrate nel database virtuale.

Per utilizzare tale funzione è necessario convalidare l'ambiente prima di esportarlo in modo da prevenire possibili inconsistenze. A tal fine, utilizzare il comando **File** → **Valida ambienti...** del menu principale oppure fare clic sul pulsante  nella barra degli strumenti.

*Esempio:*

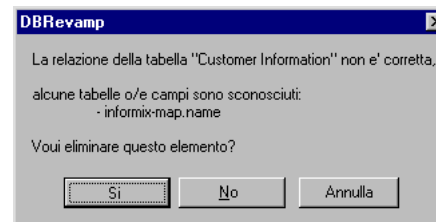
*Si consideri ad esempio il caso in cui la tabella "parameters:1" e i relativi campi siano stati copiati nell'ambiente "Marketing". Ciascun campo della tabella virtuale così creata ha come origine la tabella "parameters:1".*



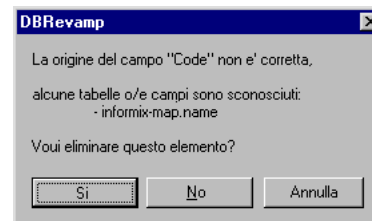
*Se successivamente questa tabella logica viene cancellata dal database reale, i campi della tabella virtuale non avranno più alcuna sorgente. In modo analogo tutte le tabelle le cui relazioni contengono un riferimento alla tabella "parameters:1" daranno origine a un'incostistenza.*

Quando riceve una richiesta di convalida di un ambiente il programma **Tun DB Revamp** procede nel modo seguente:

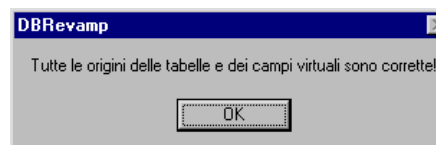
- Indica le tabelle virtuali in cui sono presenti una o più relazioni che fanno riferimento alla tabella eliminata e propone di eliminarle. In questo caso è preferibile non eliminare le relazioni ma solo i campi la cui origine si trova nella tabella eliminata.



- Indica i campi virtuali la cui origine si trova nella tabella eliminata e propone di eliminarli.




Se non è stata rilevata alcuna inconsistenza verrà visualizzata la finestra seguente:



## Esportazione di ambienti di sorgenti di dati

Per rendere disponibile agli utenti il database ridefinito in base agli ambienti è necessario esportarlo dal PC al server.

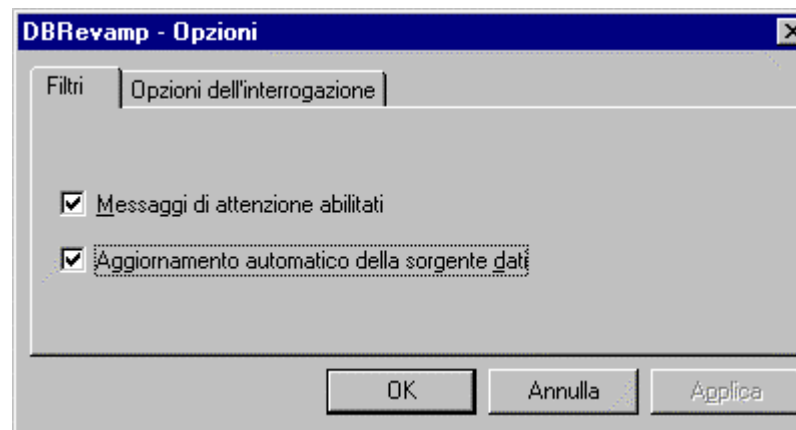
Per esportare gli ambienti delle sorgenti dei dati dal PC al server scegliere il comando **File→Esporta...** dal menu principale oppure fare clic nella barra degli strumenti sul pulsante .

Tale operazione crea o aggiorna le tre tabelle contenenti le informazioni di ridefinizione del database. Consultare al riguardo il capitolo "Revamping in Tun SQL".

## Aggiornamento di una sorgente di dati virtuale

Se il nome dell'ambiente viene modificato è possibile aggiornare la sorgente di dati virtuale corrispondente. Tale operazione avviene in modo automatico se in precedenza è stata attivata la casella **Aggiornamento automatico della sorgente dati** nella finestra **Visualizza→Opzioni**.

Verrà visualizzata la finestra di dialogo seguente:



Se tale opzione è stata selezionata sono presenti due possibilità:

- Se la casella **Messaggi di attenzione abilitati** è stata attivata, il programma **Tun DB Revamp** richiederà conferma prima di procedere all'aggiornamento automatico.
- Se la casella non è stata attivata, l'aggiornamento verrà eseguito in modo automatico senza richiedere alcuna conferma da parte dell'utente.

Se invece l'opzione **Aggiornamento automatico della sorgente dati** non è selezionata, utilizzare il comando del menu principale **Inserisci►Crea/aggiorna sorgente dati** oppure il comando del menu contestuale **Aggiorna sorgente dati** per aggiornare la sorgente di dati relativa all'ambiente in questione

Per interrompere temporaneamente il collegamento tra un ambiente e la relativa sorgente dati utilizzare il comando **Inserisci►Rimuovi sorgente dati** oppure il comando del menu contestuale **Rimuovi sorgente dati**. Il collegamento può essere ricreato in un secondo tempo con il comando **Aggiorna sorgente dati**.

## Creazione di sorgenti di dati virtuali

Il capitolo "**Configurazione e uso**" descrive come creare una sorgente dati virtuale con l'**Amministratore Sorgenti ODBC**.

È anche possibile utilizzare **Tun DB Revamp**. Scegliere **Crea sorgente dati...** dal menu di contesto degli ambienti.

Verrà visualizzata la finestra di dialogo seguente:



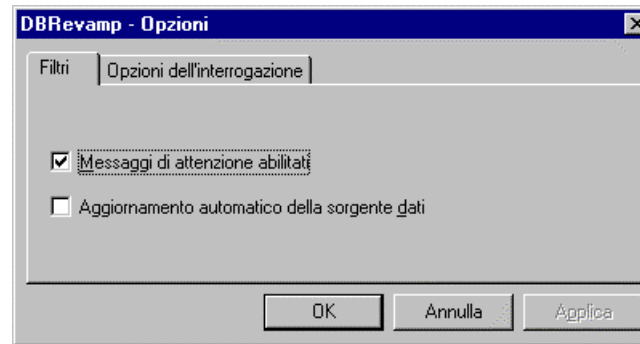
Immettere una descrizione della sorgente dati nel campo **Descrizione**. Selezionare la casella **File di traccia** e specificare il file log se si desidera tenere traccia dell'attività per la sorgente dati virtuale. Per ulteriori informazioni su questa finestra di dialogo, fare riferimento alla sezione "**Creazione di sorgenti dati virtuali**" nel capitolo "**Configurazione e uso**".

## Visualizzazione degli avvisi

L'amministratore potrà notare che durante l'uso del programma **Tun DB Revamp** viene visualizzato un certo numero di avvisi. Tali avvisi contengono vari tipi di informazioni, ad esempio su eventuali inconsistenze che potrebbero verificarsi durante l'operazione di ridefinizione del database oppure sull'assenza di alcuni elementi nella nuova struttura.




Per impostazione predefinita, il programma **Tun DB Revamp** provvede a visualizzare questi avvisi. È in ogni caso possibile disattivare la visualizzazione dei messaggi disattivando la casella **Messaggi di attenzione abilitati** (comando **Visualizza**→**Opzioni**).



## Gestione locale delle sorgenti di dati ridefinite


La descrizione del database ridefinito può essere salvata e aperta su un supporto di memorizzazione locale. Ciò risulta utile qualora l'utente non desideri esportare immediatamente il database ridefinito oppure desideri conservare copia delle versioni precedenti. La descrizione viene salvata in un file con estensione **".dbr"**.

### ➤ Memorizzazione su supporto locale

Per salvare su supporto locale la descrizione di un database ridefinito prodotto da **Tun DBRevamp**, utilizzare il comando **File**→**Salva** (o **File**→**Salva con nome...** per utilizzare un nome diverso) oppure fare clic sul pulsante  della barra degli strumenti.

Viene inoltre salvato il percorso della sorgente di dati reale; ciò consentirà in un secondo momento di esportare la sorgente di dati senza bisogno di modificare il percorso.

### ➤ Apertura di una sorgente dati locale

Per aprire una sorgente di dati ridefinita che sia salvata in un file ".dbr" sulla macchina locale, selezionare **File→Apri** dal menu principale o fare clic nella barra degli strumenti sul pulsante .

Selezionare la sorgente dati ridefinita desiderata (un file ".dbr").

È possibile aprire dal menu **File** le sorgenti dati utilizzate più di recente.

### ➤ Ricaricamento della struttura di un database






Quando si apre un file ".dbr" salvato su supporto locale è possibile aggiornare la struttura del database in cui sono stati creati gli ambienti. Ciò è utile se il database reale è stato modificato dopo l'ultimo salvataggio del file .dbr. Per fare ciò scegliere **File →Reload database structure** dal menu principale.

Dopo tale operazione è consigliabile convalidare gli ambienti creati in precedenza, specialmente se i campi reali utilizzati per definire i campi virtuali sono stati spostati o eliminati. Per ulteriori informazioni vedere la sezione "**Convalida di un ambiente**".

## Identificazione dei campi

### ➤ Icone dei campi

**Tun DB Revamp** utilizza per i campi delle icone per facilitare la lettura delle tabelle reali o virtuali.

Icone	Significato
	campo carattere
	campo data
	campo numerico
	campo binario
	chiave primaria tabella

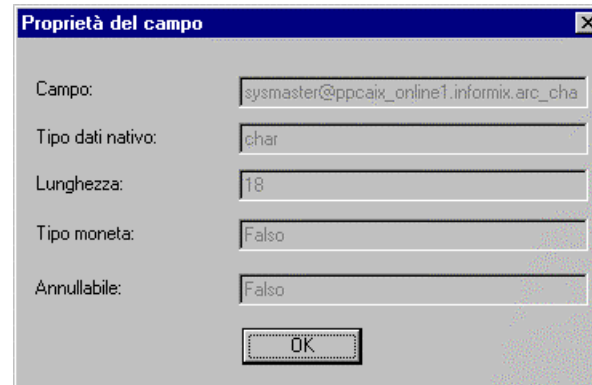




## ➤ Proprietà campi reali

Per ulteriori informazioni sul campo scegliere **Proprietà** dal menu contestuale di un campo di una tabella reale (riquadro sinistro).

Verrà visualizzata la finestra di dialogo **Proprietà del campo**:



### **Nota**

Il tipo di campo a cui si fa riferimento corrisponde al tipo nativo e quindi dipende dal DBMS utilizzato.



**PARTE 3**  
**APPENDICI**



## RIFERIMENTO

---

### Indice

**Nota:**

L'estensione xxx indica il tipo di database:

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

<b>config.XXX</b>	File con i parametri operativi e di protezione del programma server <b>Tun SQL UNIX</b> .
<b>DBMAP</b>	Applicazione Windows per creare o modificare le tabelle di conversione caratteri.
<b>DBSCRIPT</b>	Applicazione Windows che interpreta ed esegue i file batch SQL.
<b>DBSHOW</b>	Applicazione Windows per il test e la configurazione.
<b>param.XXX</b>	File con i parametri di avvio del server <b>Tun SQL UNIX</b> .
<b>tunodbc200.XXX</b>	Programma server <b>Tun SQL UNIX</b> .

## CONFIG.XXX

---

File con i parametri operativi e di protezione del programma server **Tun SQL UNIX**.

### ➤ Descrizione

I file **config.XXX** contengono un certo numero di parametri del server **Tun SQL UNIX**. Diversamente dai file **param.XXX**, questi parametri non riguardano il funzionamento generale ma un database specifico. Di seguito viene dato un esempio di file **config**.

```
#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]
```

Il file conterrà un numero di sezioni pari al numero di database gestiti dal DBMS associato al programma server. Non è necessario includere i database per i quali non siano richiesti parametri particolari.

Ogni sezione indica il database a cui si riferisce tra parentesi quadre (ad esempio [**tunsqldemo**]). Di seguito i parametri specificabili in ciascuna sezione.



**Define=END\_VARIABLE:value**

Assegna il **value** alla variabile di ambiente **ENV\_VARIABLE** (directory di installazione, formato data, ecc.) prima di aprire il database. Può ricorrere più volte nel file.

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

Alcune applicazioni per l'ufficio danno all'utente la possibilità di creare manualmente le richieste SQL secondo le proprie esigenze. In altri casi le applicazioni leggono una tabella per intero prima di visualizzarla. Ciò non rappresenta un problema per tabelle di dimensioni contenute, ma può essere un problema insormontabile nel caso di grandi quantità di dati residenti su database remoti. In questo caso infatti il numero di transazioni di rete aumenta enormemente e la memoria del PC potrebbe essere insufficiente a contenere i dati ricevuti, con conseguente necessità di riavviare il PC dopo un'operazione di "select". Per ovviare a questo problema, è possibile impostare dei limiti applicati dal driver ODBC con il parametro **RowLimitMode**. Il valore assegnato al parametro ignora altri valori eventualmente definiti nell'Origine dati sul PC.

Vi sono cinque tipi di limite:

<b>None</b>	Il driver non imposta alcun limite.
<b>Absolute</b>	Il driver non carica più di <b>RowLimitValue</b> righe per operazione <b>select</b> . L'utente non viene avvisato.
<b>Fixed</b>	Il driver non carica più di <b>RowLimitValue</b> righe per operazione <b>select</b> . L'utente viene informato con un messaggio.
<b>Variable</b>	Il driver non carica più di <b>RowLimitValue</b> righe per operazione <b>select</b> . Un messaggio comunque propone di caricarne altre senza superare il limite indicato da <b>RowLimitMax</b> .
<b>Extended Limit</b>	Il driver non carica più di <b>RowLimitValue</b> righe per operazione <b>select</b> . Un messaggio comunque propone di caricarne altre senza un limite Massimo. Il messaggio visualizzato ha solo valore informativo.

### **DbmsName=DatabaseName**

Imposta o cambia il nome del database da trasmettere al driver ODBC.

### **Version=DatabaseVersion**

Imposta o cambia la versione del database da trasmettere al driver ODBC.

Oltre alle sezioni corrispondenti a ciascun database, il file **config** può comprendere le sezioni **[Allowed]** e **[Denied]** utilizzate per proteggere l'accesso di alcuni database. Le funzioni operano nel modo seguente:

#### **[Allowed]**

Questa sezione contiene una serie di tre elementi:

**base\_name , user\_name , product\_name**

dove

- **base\_name** è il nome del database;
- **user\_name** è il nome dell'utente del database;
- **product\_name** è il nome dell'applicazione Windows che dialoga con il server.

Ciascuna serie indica che il **user\_name** è autorizzato all'accesso del **base\_name** tramite l'applicazione **product\_name**. Un parametro può essere sostituito da \*.

Ad esempio, la serie **tunsql Demp,\* , excel** indica che tutti gli utenti possono utilizzare il database **tunsqldemo** con l'applicazione Excel, tranne gli utenti indicati in modo analogo nella sezione **[Denied]**.

#### **[Denied]**

Contiene una serie di tre elementi come nella sezione **[Allowed]**. Ciascuna serie indica che il **user\_name** non è autorizzato all'accesso del **base\_name** tramite l'applicazione **product\_name**. Un parametro può essere sostituito da \*.

Ad esempio, la serie **\* ,gianni ,excel** indica che l'utente **gianni** non può utilizzare alcun database tramite **excel**, operazione invece consentita agli utenti indicati nella sezione **[Allowed]**.





**Nota:**

Per utilizzare i parametri di un file **config**, avviare il programma server **Tun SQL UNIX** con l'opzione **-c** sulla riga di comando.

È possibile far coesistere due diversi motori di database Informix (Informix versione 5 e Informix versione 7). Ad esempio, il motore 1 della directory /u/informix1 può accedere ad un database mentre il motore 2 della directory /u/informix2 può accedere ad un altro database. In questo caso il file config.ifx conterrà:

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

➤ **Vedere inoltre**

param.XXX, tunodbc.XXX

## DBMAP

---

Applicazione Windows per la creazione e la modifica delle tabelle di conversione caratteri.

### ➤ Sintassi

**DBMAP [-ffile]**

### ➤ Descrizione

**Tun DB Map** serve per creare e modificare le tabelle di conversione caratteri. Le tabelle garantiscono la compatibilità dei caratteri accentati e altri segni diacritici sui PC e sui sistemi DBMS remoti in quanto questi utilizzano codici di carattere diversi. Dichiarare le tabelle di conversione nella definizione dell'Origine dati.

#### **-ffile**

Il file contenente una tabella di conversione.



## DBSCRIPT

---

Applicazione Windows che interpreta ed esegue file batch SQL.

### ➤ Sintassi

```
DBSCRIPT [-dorigine_dati] [-ffile]
```

### ➤ Descrizione

**Tun DB Script** consente di eseguire con un'unica operazione una lista di richieste SQL interpretando i comandi in modo sequenziale e fermandosi eventualmente in presenza di errori. **Tun DB Script** può inoltre essere utilizzato per modificare e salvare file batch SQL.

**Tun DB Script** serve a caricare il contenuto di un database da un PC a un DBMS remoto in ambiente Windows e per l'aggiornamento di grandi quantità di dati o per la cancellazione di database

#### **-ffile**

Il file contenente i comandi SQL caricati all'avviamento dell'applicazione.

#### **-dorigine\_dati**

L'Origine dati elaborata dal file batch SQL. La connessione va stabilita manualmente dall'utente.

## DBSHOW

---

Applicazione Windows per il test e la configurazione.

### ➤ Sintassi

`DBSHOW [-hhost]`

### ➤ Descrizione

Interroga un host UNIX della rete per verificare la presenza o meno di uno o più server **Tun SQL** UNIX.

Indicare il nome dell'**Host** e fare clic su **Ricerca server**. Se sul sistema sono installati correttamente uno o più programmi server ne verrà indicato il nome e il nome dei DBMS associati.

Questo programma serve soprattutto per verificare la correttezza dell'installazione.

#### **-hhost**

Il nome dell'host a cui inviare la richiesta.



## PARAM.XXX

---

File con i parametri di configurazione del server **Tun SQL** UNIX.

### ➤ Descrizione

Per evitare di specificare lunghe sequenze di opzioni sulla riga di comando del programma server **Tun SQL** UNIX, è consigliabile creare un file contenente le opzioni e indicare questo file come opzione **-f** sulla riga di comando.

La procedura di installazione di **Tun SQL** provvede automaticamente alla creazione di questi file (file **param.XXX**, ovvero **param.ora**, **param.syb**, **param.ifx** a seconda del DBMS a cui si riferiscono).

Di seguito un file **config.XXX** di esempio.

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

Per il significato delle opzioni, vedere la sezione **tunodbc200.XXX**.

### ➤ Progress

Alcuni campi Progress possono contenere più valori: questi sono conosciuti come "array fields". Per visualizzare questi valori da applicazioni tipo MS Query e MS Access, è necessario inserire la seguente opzione nel file param.proX (dove X è il numero della versione di Progress utilizzato): :

```
-arrayfields=*
```

dove \* sta per uno dei seguenti caratteri:

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

Il carattere predefinito è \_.

Le colonne richieste per visualizzare i diversi campi array verranno quindi chiamate:  
columnname\*n\*,  
dove \* è il carattere scelto nel file param.proX (valore predefinito, "\_")  
e n è la posizione del valore nella tabella.

*Esempio:*

*Il secondo valore nel campo array appare nella colonna col\_2\_.*

Se l'utilizzo del carattere "\_" causa un problema quando la colonna viene generata (altre colonne potrebbero avere un nome simile), è necessario scegliere uno degli altri quattro caratteri.

### ➤ **Vedere inoltre**

config.XXX, tunodbc.XXX



## TUNODBC200.XXX

---

Programma server UNIX di **Tun SQL**.

**Nota:**

I diversi server UNIX di **Tun SQL** hanno alcune opzioni in comune. L'elenco delle opzioni può essere visualizzato avviando l'eseguibile **tunodbc200.xxx** con l'opzione **-a[ll]**, dove xxx indica il tipo di database utilizzato.

### ➤ Sintassi

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

## ➤ Descrizione

### **-a**

Elenca le opzioni supportate da **tunodbc200.XXX**.

### **-c=file\_config**

Associa un file di configurazione (**config.XXX**) al server (vedere sezione **config.XXX**).

### **-db=name**

Associa il nome del DBMS al programma. Il nome viene visualizzato nell'elenco avviando **Tun DB Show**.

### **-de**

Attiva la modalità di tracciamento del programma. I messaggi vengono visualizzati per impostazione predefinita sulla periferica **/dev/console**.

### **-Dname=value**

Imposta il **value** della variabile di ambiente **dname** (directory di installazione del database, formato dei dati, ecc.) prima di avviare il programma server. Può essere ripetuta più volte sulla riga di comando. Alcuni DBMS richiedono necessariamente la definizione di alcune variabili per il funzionamento del programma, operazione comunque svolta dalla procedura di installazione. I DBMS che richiedono la definizione delle variabili sono:

### **Oracle**

ORACLE\_HOME: Directory di installazione.

ORACLE\_SID: Database predefinito.

### **Informix**

INFORMIX\_DIR: Directory di installazione.

DBPATH: Directory contenente il database (solo SE).





## Sybase

SYBASE: Directory di installazione.

SYBSERVNAME: File di configurazione del server (opzionale).

Equivale alla variabile DSQUERY definita e utilizzata da SYBASE.

### **-f=param\_file**

Il nome di un file contenente in sequenza le opzioni di avvio del programma. Risparmia la digitazione delle opzioni sulla riga di comando.

### **-i**

Modalità interattiva di test per verificare il corretto funzionamento del server.

### **-o=file**

Il file o la periferica in cui vengono memorizzati i messaggi di tracciamento del programma. Funziona solo in modalità **debug**.

### **-o2=file**

Il file o la periferica in cui vengono memorizzati i messaggi di tracciamento Sentinella. Funziona solo in modalità **debug**.

### **-t=xx**

Il tempo di timeout trascorso il quale l'assenza di risposta da parte del PC viene interpretata come un arresto del sistema PC. In questo caso il programma server termina l'esecuzione. Il valore ha comunque priorità inferiore al valore assegnato sul PC quanto è stata definita l'Origine dati.

### **-u**

Con il parametro che segue definisce l'elenco degli utenti autorizzati ("\*" per tutti). Valore predefinito: \*. Ad esempio:

**-x = \***

**-u = bill** (solo "bill" è autorizzato)

### **-v=XX**

Associa la versione del DBMS al programma. La versione viene visualizzata in un elenco avviando **Tun DB Show**.

### **-x**

Definisce l'elenco degli utenti non autorizzati all'accesso ("\*" per tutti). Ad esempio:

```
-u = *  
-x = bill    (solo "bill" non è autorizzato)
```

## ➤ **Opzioni Informix**

### **-h**

Per impostazione predefinita, Informix non mantiene aperti i cursori durante l'esecuzione dei comandi **commit** e **rollback**. L'opzione **-h** assicura che i cursori rimangano aperti dopo uno di questi comandi se le applicazioni che utilizzano questo tipo di server non sono in grado di farlo.

### **-n**

Vale solo per SCO UNIX 5. Disabilita il controllo delle password utenti se il sistema SCO UNIX 3.2 versione 5 non effettua correttamente tale controllo.

### **-s**

Per impostazione predefinita, Informix non esegue i comandi **select** con un'opzione di ordinamento (**group by** o **order by**) su una colonna non inclusa nel comando **select**. L'opzione **-s** può essere utilizzata per ovviare a questa mancanza qualora le applicazioni non vi provvedano autonomamente.



## ➤ Opzioni Oracle

### **-l**

Se nel catalogo sono stati creati tabelle, colonne, indici o viste contenenti caratteri minuscoli, l'opzione -l assicura che le funzioni del catalogo restituiscano i dati racchiusi tra virgolette. Le applicazioni che utilizzano questo server creeranno query in cui i nomi sono racchiusi tra virgolette.

## ➤ Opzioni Progress

### **-n**

Vale solo per SCO UNIX 5. Disabilita il controllo delle password utenti se il sistema SCO UNIX 3.2 versione 5 non effettua correttamente tale controllo .

### **-nor**

Il database Progress non è in grado di determinare il numero di righe modificate o eliminate con un comando **update** o **delete**. Il server Progress è preimpostato per risolvere questo inconveniente, anche se ciò comporta un rallentamento ad ogni comando **update** o **delete**. Se le applicazioni che utilizzano il server non hanno bisogno dei dati relativi a righe modificate o eliminate, utilizzare l'opzione **-nor** per disabilitare la funzione del server e ottenere così un miglioramento delle prestazioni.

### **-ow**

Per impostazione predefinita il server non supporta il concetto di proprietario di un oggetto nel database. Alcune applicazioni infatti tentano di aggiungere un prefisso indicante il proprietario quando con gli oggetti viene indicato il proprietario provocando in tal modo errori durante l'esecuzione. Questa opzione consente una corretta gestione delle informazioni sui proprietari qualora l'applicazione richieda queste informazioni.

### **-p=XX**

Se è necessario utilizzare le opzioni specifiche del database Progress (ad esempio l'opzione -Q, che rende il database compatibile con lo standard ANSI), utilizzare -p=XX, dove **XX** va sostituito con una stringa contenente le opzioni richieste tra virgolette.

### **-sv**

Il database Progress utilizza solo un tipo di stringa caratteri. Per impostazione predefinita, tutte le stringhe vengono considerate di lunghezza fissa (SQL\_CHAR in ODBC). Se viene utilizzata questa opzione, le stringhe verranno trattate come se avessero lunghezza variabile (SQL\_VARCHAR in ODBC).

### **-sy**

Il database Progress può definire colonne di sistema o nascoste non restituite da un'operazione di ricerca con caratteri jolly effettuata su tutte le colonne. È per questo motivo che, per impostazione predefinita, il server non descrive le colonne nel proprio catalogo. L'opzione comporta la restituzione delle colonne nascoste qualora queste si rendessero necessarie.

## ➤ **Vedere inoltre**

param.XXX, config.XXX



# COMANDI SQL UTILIZZATI IN C-ISAM

---

## Istruzioni principali

CREATE DATABASE .....	103
CREATE TABLE .....	104
DEFINE TABLE.....	105
COLUMN DEFINITION OPTION .....	106
DEFAULT CLAUSE .....	107
NOT NULL CLAUSE.....	108
CONSTRAINT DEFINITION SUBSET.....	109
CONSTRAINT DEFINITION OPTION.....	110
FILE IS OPTION .....	111
CREATE INDEX.....	112
CREATE SYNONYM .....	113
COMMENT .....	114
DROP DATABASE.....	115
CONNECT DATABASE.....	116
DISCONNECT DATABASE .....	117
DROP INDEX.....	118
DROP TABLE .....	119

DROP SYNONYM.....	120
UNDEFINE TABLE.....	121
SELECT.....	122
SELECT CLAUSE.....	123
EXPRESSION.....	124
FROM CLAUSE.....	125
WHERE CLAUSE.....	126
GROUP BY CLAUSE.....	127
HAVING CLAUSE.....	128
ORDER BY CLAUSE.....	129
DELETE.....	130
INSERT.....	131
VALUES CLAUSE.....	132
UPDATE.....	133
SET CLAUSE.....	134
AGGREGATE EXPRESSION.....	135

## Sintassi di comando SQL

I comandi SQL sono presentati utilizzando la seguente sintassi:

- I nomi riservati sono in caratteri maiuscoli (INSERT, UNIQUE, etc.). É comunque possibile utilizzare caratteri minuscoli per inserire questi nomi nella linea di comando.
- I nomi di variabile sono in corsivo (*Databasename, ecc.*).
- Le parentesi quadre indicano parametri o valori opzionali ([opzionale]).
- Parentesi graffe e l'espressione **xor** indica una scelta mutualmente esclusiva ({ A xor B xor C }).
- L'esponente n (<sup>n</sup>) indica una sequenza che può essere ripetuta da 0 a n volte (sequenza<sup>n</sup>).

I segni di punteggiatura e le parentesi sono simboli letterali che devono essere digitati esattamente come appaiono.



## CREATE DATABASE

---

### ➤ Funzione

Crea un nuovo database.

### ➤ Sintassi

```
CREATE DATABASE Nomebase
```

**Nota:**

Il nome del database deve essere inferiore a 18 caratteri.

### ➤ Utilizzo

Il database creato diventa il database corrente.

Questo comando può essere utilizzato soltanto con **sqltools** (Strumento di **Tun SQL**).

Dopo la creazione del database, viene creata una directory chiamata *nomeDatabase.ism*. Questa directory contiene i file C-ISAM aggiuntivi che compongono il catalogo (*SysTables*, *SysColumns*, *SysIndexes*, *SysDefaults*). È una subdirectory della directory definita dalla variabile di ambiente *ISAM-PATH* se questa è definita, oppure della directory corrente.

### ➤ Esempio

```
CREATE DATABASE TEST;
```

Crea la directory *test.ism* contenente i file *SysTables.dat*, *SysTables.idx*, *SysColumns.dat*, *SysColumns.idx*, *SysIndexes.dat*, *SysIndexes.idx*, *SysDefaults.dat*, e *SysDefaults.idx*.



## CREATE TABLE

---

### ➤ Funzione

Crea una nuova tabella nel database corrente e pone i vincoli di integrità dati sulle colonne e sui gruppi di colonne.

### ➤ Sintassi

```
CREATE TABLE tablename (Column definition [,Column definition]n)  
[,Constraint definition]n
```

**Nota:**

Il nome tabella deve essere inferiore a 18 caratteri.

### ➤ Utilizzo

I nomi tabella nello stesso database devono essere unici. Ogni colonna nella stessa tabella deve avere un nome differente.

Il nome tabella deve essere prefissato con il nome di un utente UNIX che diventa il proprietario della tabella. Se non è specificato nessun nome viene utilizzato l'ID di login corrente.

### ➤ Esempio

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

Questo comando crea la tabella *table1* nel database relazionale creando i file C-ISAM associati (*table1\_100.dat* e *table1\_100.idx*, per esempio).

I nomi file C-ISAM sono creati prendendo i primi sette caratteri del nome tabella ed aggiungendogli un valore unico. Se il nome tabella ha meno di sette caratteri, i nuovi nomi file vengono completati con il segno di sottolineato ( \_ ). Il valore unico è 100 per la prima tabella creata: questo valore viene incrementato di 1 per ogni nuova tabella.





## DEFINE TABLE

---

### ➤ Funzione

Definisce una nuova tabella nel database corrente, con vincoli di integrità sulle colonne o su sequenze di colonne, ed opzioni condizionali per l'esistenza dei file.

### ➤ Sintassi

```
DEFINE TABLE tablename [File is option] (Column definition  
[,Column definition]n) [,Constraint definition]n
```

**Nota:**

Il nome di tabella deve essere inferiore a 18 caratteri.

### ➤ Utilizzo

I nomi di tabella nello stesso database devono essere unici. Ogni colonna nella stessa tabella deve avere un nome differente.

Il nome tabella deve essere prefissato con il nome di un utente UNIX che diventa il proprietario della tabella. Se non è specificato nessun nome viene utilizzato l'ID di login corrente.

### ➤ Esempio

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

In questo esempio, i file C-ISAM *file\_1* (*file\_1.idx* e *file\_1.dat*) esistono già: deve quindi essere definita soltanto la tabella.

## COLUMN DEFINITION OPTION

---

Utilizzato nelle comandi **CREATE TABLE** e **DEFINE TABLE**.

### ➤ Funzione

L'opzione "column definition" nel comando **DEFINE TABLE** (**CREATE TABLE**) definisce il nome, il tipo, il valore predefinito ed il vincolo per una singola colonna.

### ➤ Sintassi

*Columnname* Data type [Default clause] [Not null clause] [Constraint definition option]

### ➤ Esempio

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

Questa tabella è composta dalle colonne *name*, *race* e *sex*.



## DEFAULT CLAUSE

---

Utilizzato nell'opzione **COLUMN DEFINITION**.

### ➤ Sintassi

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

### ➤ Utilizzo

Il valore predefinito è inserito nella colonna quando non viene specificato un valore esplicito. Se non è specificato un valore predefinito è la colonna consente valori nulli, il predefinito diventa NULL.

<b>LITERAL</b>	Stringa di caratteri o di costanti numeriche definite dall'utente
<b>NULL</b>	Valore Null
<b>CURRENT</b>	Data e ora corrente (utilizzabile soltanto con tipo TIMESTAMP)
<b>TODAY</b>	Data corrente (utilizzabile soltanto con tipo DATE)
<b>USER</b>	Nome dell'utente corrente (utilizzabile soltanto con tipo VAR VARCHAR)

### ➤ Esempio

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

In questa tabella, il valore predefinito per *sex* è una stringa literal: 'M'.



Utilizzato nell'opzione **COLUMN DEFINITION**.

### ➤ Funzione

Se non viene indicato un valore predefinito per una colonna, viene assunto il valore null a meno che dopo il tipo dato della colonna venga inclusa la parola chiave NOT NULL. In questo caso, non c'è valore predefinito per la colonna.

### ➤ Sintassi

NOT NULL

### ➤ Esempio

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

Se la colonna è definita come NOT NULL (e non è specificato un valore predefinito), è necessario immettere un valore in questa colonna durante l'inserimento di una riga o l'aggiornamento di questa colonna in una riga. Diversamente, il server database ritorna un errore.



## CONSTRAINT DEFINITION SUBSET

---

Utilizzato nell'opzione **COLUMN DEFINITION**.

### ➤ Funzione

Il vincolo "constraint definition" consente la creazione di vincoli per una singola colonna.

### ➤ Sintassi

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

**Nota:**

Il nome vincolo non deve superare i 18 caratteri e deve essere unico nel database.

### ➤ Utilizzo

<b>UNIQUE</b>	Vincola il campo ad essere unico
<b>PRIMARY KEY</b>	Vincola il campo ad essere unico e ad essere la chiave primaria della tabella database

Se il nome del vincolo non è specificato, viene assegnato un valore predefinito.

### ➤ Esempio

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

Il vincolo di unicità del numero fattura è chiamato *un\_invoice*.



## CONSTRAINT DEFINITION OPTION

---

Utilizzato nelle comandi **CREATE TABLE** e **DEFINE TABLE**.

### ➤ Funzione

Le opzioni di definizione vincoli consente di creare vincoli per un gruppo di colonne (da 1 a 8 colonne).

### ➤ Sintassi

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

### ➤ Utilizzo

<b>UNIQUE</b>	Vincola il campo ad essere unico per il gruppo di colonne
<b>PRIMARY KEY</b>	Vincola il campo ad essere unico e ad essere la chiave primaria per il gruppo di colonne

Se il nome del vincolo non è specificato, viene assegnato al vincolo un nome predefinito.

Ogni colonna nominata in un vincolo deve essere una colonna nella tabella e non può apparire più di una volta nella lista dei vincoli.

### ➤ Esempio

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

Il vincolo di chiave primaria *pk\_family* riguarda i campi *name* e *surname*.



Utilizzato nel comando **DEFINE TABLE**.

### ➤ Funzione

Definisce l'utilizzo della tabella secondo l'esistenza o la non esistenza di un file.

### ➤ Sintassi

FILE IS *filename*

### ➤ Utilizzo

Se viene utilizzata questa opzione non vengono creati file C-ISAM. **Tun SQL** deve trovare i file *filename.dat* e *filename.idx* nella directory del database corrente prima di poter utilizzare la tabella.

Se l'opzione non viene utilizzata, ai file creati nella directory del database corrente viene assegnato un nome predefinito.

### ➤ Esempio

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

In questo esempio non viene creato nessun file. I file *foo1.dat* e *foo1.idx* devono essere aggiunti nella directory *test.ism*.

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

In questo esempio, i file *table\_100.dat* e *table\_100.idx* (nomi predefiniti) vengono creati nella directory *test.ism*.

## CREATE INDEX

---

### ➤ Funzione

Crea un indice per una o più colonne nella tabella (da 1 a 8 colonne).

### ➤ Sintassi

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

**Nota:**

Il nome indice non deve superare i 18 caratteri. Il numero delle colonne può variare da 1 a 8.

### ➤ Utilizzo

<b>UNIQUE</b>	Vincola il campo ad essere unico
<b>DISTINCT</b>	Sinonimo di UNIQUE

Per tabelle definite con l'opzione FILE IS, il comando CREATE INDEX deve essere eseguito prima di copiare i file filename.dat e filename.idx.

### ➤ Esempio

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

Questo comando crea l'indice *ix\_name* per le colonne *name* e *birth\_date* nella tabella *Table1*.





## CREATE SYNONYM

---

### ➤ Funzione

Assegna un sinonimo ad una tabella.

### ➤ Sintassi

```
CREATE SYNONYM synonymname FOR tablename
```

#### Nota

Il nome del sinonimo non può superare i 18 caratteri.

### ➤ Utilizzo

Il nome del sinonimo deve essere preceduto dal nome di un utente UNIX che viene così configurato quale proprietario del sinonimo. Se non viene indicato alcun nome verrà utilizzato l'ID di login corrente.

### ➤ Esempio

```
CREATE SYNONYM employee FOR Table1;
```

L'istruzione crea il sinonimo *employee* per la tabella *Table1*.

## COMMENT

---

### ➤ Funzione

Associa una nota ad una tabella o ad un sinonimo.

### ➤ Sintassi

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

### ➤ Esempio

```
COMMENT on TABLE1 IS 'Tabella dipendenti'
```



## DROP DATABASE

---

### ➤ Funzione

Rimuove (cancella) un intero database, inclusi tutti i cataloghi, gli indici e i dati.

### ➤ Sintassi

DROP DATABASE *basename*

**Nota:**

Il nome database dev'essere inferiore a 18 caratteri.

### ➤ Utilizzo

Un database utilizzato da altri utenti non può essere rimosso.

Il comando DROP DATABASE non rimuove la directory del database se questa contiene altri file oltre a quelli creati per le tabelle e gli indici del database.

### ➤ Esempio

```
DROP DATABASE DBTEST;
```

Cancella il database *DBTEST*.

## CONNECT DATABASE

---

### ➤ Funzione

Si connette ad un database differente. Non è possibile effettuare operazioni su un database se non ci si è connessi.

### ➤ Sintassi

```
CONNECT DATABASE basename
```

### ➤ Esempio

```
CONNECT DATABASE DBTEST2;
```

Si connette al database *DBTEST2*.



## DISCONNECT DATABASE

---

### ➤ Funzione

Si disconnette dal database corrente.

### ➤ Sintassi

DISCONNECT DATABASE *basename*

### ➤ Esempio

```
DISCONNECT DATABASE DBTEST2;
```

Si disconnette dal database *DBTEST2*.



## DROP INDEX

---

### ➤ Funzione

Cancella un indice.

### ➤ Sintassi

DROP INDEX *indexname*

### ➤ Esempio

```
DROP INDEX ix_name ;
```

Cancella l'indice *ix\_name*.



## DROP TABLE

---

### ➤ Funzione

Cancella una tabella, compresi i dati e gli indici a questa associati.

### ➤ Sintassi

```
DROP TABLE {tablename xor synonymname}
```

### ➤ Utilizzo

Se un sinonimo viene eliminato dall'istruzione **DROP TABLE**, verrà eliminata anche la tabella.

Se l'istruzione **DROP TABLE** viene applicata ad una tabella, i sinonimi della tabella verranno eliminati solo utilizzando l'istruzione **DROP SYNONYM**.

### ➤ Esempio

```
DROP TABLE TABLE1;
```

Cancella la tabella TABLE1, e i suoi indici e i dati.

## DROP SYNONYM

---

### ➤ Funzione

Cancella un sinonimo precedentemente definito.

### ➤ Sintassi

DROP SYNONYM *synonymname*

### ➤ Utilizzo

Se viene cancellata una tabella, il sinonimo rimane fino a quando verrà esplicitamente rimosso con il comando DROP SYNONYM.

### ➤ Esempio

```
DROP SYNONYM employee;
```

Elimina il sinonimo *employee* assegnato alla tabella *Table1*. Con questa istruzione la tabella non viene eliminata.





## UNDEFINE TABLE

---

### ➤ Funzione

Cancella una tabella definita da un comando DEFINE senza cancellare i dati e gli indici associati.

### ➤ Sintassi

```
UNDEFINE TABLE {tablename xor synonymname}
```

### ➤ Esempio

```
UNDEFINE TABLE TABLE1;
```

Cancella la tabella TABLE1 creata dal comando DEFINE TABLE TABLE1.

## SELECT

---

### ➤ Funzione

Interroga un database.

### ➤ Sintassi

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

### ➤ Utilizzo

É possibile interrogare le tabelle nel database corrente o in uno diverso.

### ➤ Esempio

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

Questa interrogazione estrae i clienti con un fatturato (turn\_over) annuo superiore a 250 dalla tabella *customers* e ne ordina i nomi (customer\_name) per paese (country).



## SELECT CLAUSE

---

Utilizzato nelle comandi **SELECT** e **INSERT**.

### ➤ Sintassi

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

Nell'opzione **SELECT**, è necessario specificare quali dai selezionare, e se si desidera omettere i valori duplicati.

### ➤ Utilizzo

<b>ALL</b>	Specifica che siano estratti tutti i valori selezionati, anche se duplicati.
<b>DISTINCT</b>	Rimuove le righe duplicate dai risultati dell'interrogazione.
<b>UNIQUE</b>	Sinonimo per <b>DISTINCT</b> .

### ➤ Esempio

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

Questa interrogazione estrae i clienti con un fatturato (*turn\_over*) annuo superiore a 250 dalla tabella *customers* e ne ordina i nomi (*customer\_name*) per paese (*country*).

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

Questa interrogazione ritorna la data dell'ordine (*order\_date*), il numero degli ordini (**COUNT**) e la differenza tra la data di pagamento (*paid\_date*) e la data dell'ordine (*order\_date*), raggruppati per data dell'ordine e intervallo di tempo (differenza tra le date).

Utilizzato nel comando **SELECT**.

### ➤ Sintassi

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

### ➤ Esempio

```
'Cordwainer'
```

La stringa caratteri *'Cordwainer'* è una sub-espressione.



## FROM CLAUSE

---

Utilizzato nelle comandi **SELECT** e **DELETE**.

### ➤ Funzione

Elenca la(e) tabella(e) da cui i dati vengono selezionati.

### ➤ Sintassi

```
FROM {tablename xor synonymname } [[AS] tablealias]  
    [{tablename xor synonymname } [[AS] tablealias]]n
```

### ➤ Esempio

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

In questo comando, i dati vengono estratti dalle tabelle *customers* e *orders* e alle tabelle vengono assegnati degli alias.



## WHERE CLAUSE

---

Utilizzato nelle comandi **SELECT**, **DELETE** e **UPDATE**.

### ➤ Funzione

Specifica le condizioni di ricerca e di unione (join) per i dati selezionati.

### ➤ Sintassi

WHERE Condition [AND Condition]<sup>1</sup>

### ➤ Esempio

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

In questo esempio, la condizione di ricerca è la “data ultimo ordine”.



## GROUP BY CLAUSE

---

Utilizzato nel comando **SELECT**.

### ➤ Funzione

Produce una singola riga di risultati per ogni gruppo.

### ➤ Sintassi

```
GROUP BY {tablename xor synonymname} . Column name xor Select  
number}  
[, {tablename xor synonymname } . Column name xor Select number } ]n
```

### ➤ Utilizzo

Un gruppo è un insieme di righe che hanno lo stesso valore per ogni colonna elencata.

La variabile "select number" è un intero che rappresenta la posizione di una colonna nella condizione SELECT.

### ➤ Esempio

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

I risultati sono raggruppati per *order\_date* e *paid\_date - order\_date*.

Utilizzato nel comando **SELECT**.

### ➤ **Funzione**

Applica una o più condizioni ai gruppi.

### ➤ **Sintassi**

HAVING condition

### ➤ **Esempio**

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

Questa interrogazione ritorna le tabelle call\_code, call\_dtime e customer\_num e le raggruppa per call\_code per tutte le chiamate da clienti con numero cliente (customer\_num) inferiore a 42.





## ORDER BY CLAUSE

---

Utilizzato nel comando **SELECT**.

### ➤ Funzione

Ordina i risultati dell'interrogazione per i valori contenuti in una o più colonne.

### ➤ Sintassi

`ORDER BY {tablename xor synonymname .} Column name xor Select number xor Display label [, {tablename xor synonymname .} Column name xor Select number xor Display label}]n`

### ➤ Utilizzo

La variabile "select number" è un intero che rappresenta la posizione di una colonna nella condizione SELECT.

### ➤ Esempio

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

Questa interrogazione ritorna i propri risultati ordinati per paese (country).



## DELETE

---

### ➤ Funzione

Cancella una o più righe da una tabella.

### ➤ Sintassi

```
DELETE FROM {tablename xor synonymname}  
[WHERE {Condition xor CURRENT OF Cursor name}]
```

### ➤ Esempio

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

Questo comando cancella dalla tabella *customers* le righe in cui la data dell'ultimo ordine (*last\_order\_date*) è antecedente al 1992.



## INSERT

---

### ➤ Funzione

Inserisce una o più righe in una tabella.

### ➤ Sintassi

```
INSERT INTO {tablename xor synonymname}  
[(Column name [,Column name]n)] {Values clause xor Select clause}
```

### ➤ Esempio

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

Questo comando inserisce i valori *'Socks'*, *'Cat'* e *'M'* nella tabella *Pets*.

## VALUES CLAUSE

---

Utilizzato nel comando **INSERT**.

### ➤ Sintassi

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

### ➤ Utilizzo

Quando si utilizza la condizione VALUES, è possibile inserire soltanto una riga alla volta. Ogni valore che segue la parola chiave VALUES è assegnato alla colonna corrispondente elencata nella condizione INSERT INTO (oppure alle colonne in ordine se non viene specificata una lista di colonne).

### ➤ Esempio

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

Questo comando inserisce il valore *'Socks'* nella prima colonna e *'M'* nella terza colonna della tabella *Pets*. La seconda colonna non viene toccata.



## UPDATE

---

### ➤ Funzione

Modifica il valore di una o più colonne oppure di una o più righe in una tabella.

### ➤ Sintassi

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

### ➤ Esempio

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

Questo comando imposta il prezzo di tutti gli articoli tipo 'L' nella tabella *Catalog* su 10.

## SET CLAUSE

---

Utilizzato nel comando **UPDATE**.

### ➤ Sintassi

{Column name = {Constant xor (Select statement) xor NULL}  
[,Column name = {Constant xor (Select statement) xor NULL}]<sup>n</sup>  
xor {(Column name [,Column name]<sup>n</sup> xor \*) = (Select statement)}

### ➤ Esempio

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

In questo comando la clausola **SET** imposta il prezzo *item\_price* su 10.



## AGGREGATE EXPRESSION

---

Utilizzato nel comando **SELECT** o nell'espressione.

### ➤ Sintassi

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

### ➤ Esempio

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

Questo comando ritorna il numero dei diversi tipi di articolo nella tabella *Catalog*.

## Tipi di dati

Questa sezione descrive i tipi di dati supportati dal linguaggio SQL con le equivalenti definizioni in linguaggio C. Le descrizioni corrispondono ai tipi SQL utilizzati da CREATE TABLE. Questi sono equivalenti in modo nativo. Vengono riportate le differenze per l'importazione di file creati al di fuori del database C-ISAM tramite il comando DEFINE TABLE.

Nella tabella che segue, i tipi in corsivo nella colonna "tipi in linguaggio C" sono applicabili soltanto nell'utilizzo di CREATE TABLE.

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

### ➤ Il tipo bit

Questo tipo corrisponde al tipo SQL\_BIT in ODBC. Memorizza i valori binari 0 e 1, oppure il valore null.





Se questo tipo di dato viene utilizzato con CREATE TABLE, un blocco di due byte viene riservato nel file creato per differenziare tra il valore null e 0 o 1. Se un campo del genere viene utilizzato nella definizione di una chiave, i due byte vengono usati per la chiave.

Se il tipo di dato bit viene utilizzato con DEFINE TABLE, viene riservato soltanto un byte. Il valore 0 non viene più differenziato dal valore null. In DEFINE TABLE, quindi, questo tipo di campo non può essere null.

La tabella che segue mostra i valori memorizzati. I dati in corsivo si applicano solo a CREATE TABLE.

<b>Campo tipo bit</b>	<i>notnull_data</i>	<b>Valore dati</b>
0	<i>1</i>	0
1	<i>1</i>	1
<i>Null</i>	<i>0</i>	0

### ➤ Il tipo byte

Questo tipo corrisponde al tipo SQL\_TINYINT in ODBC. Memorizza valori da 0 a 255, oppure null.

Se questo tipo di dato viene utilizzato con CREATE TABLE viene riservato un blocco di due byte (come per il tipo **bit**) per differenziare il valore null da altri valori. Se questo tipo di campo è utilizzato nella definizione di una chiave, i due byte sono usati per la chiave.

Se il tipo di dato è utilizzato con DEFINE TABLE, viene riservato soltanto un byte. Il valore 0 non può quindi essere differenziato dal valore null e pertanto in DEFINE TABLE questo tipo di campo non può essere null.

La seguente tabella mostra i valori memorizzati. I dati in corsivo si applicano soltanto a CREATE TABLE.

<b>Campo tipo byte</b>	<i>notnull_data</i>	<b>Valore dati</b>
0	<i>1</i>	0
1	<i>1</i>	1
...	<i>1</i>	...
255	<i>1</i>	255
<i>null</i>	<i>0</i>	0

### ➤ Il tipo char

Questo tipo corrisponde al tipo SQL\_CHAR in ODBC. Può memorizzare da 1 a 32511 caratteri.

Quando questo tipo di campo è inserito in un campo, tutti gli spazi inutili alla fine di una stringa vengono cancellati e i byte corrispondenti nel file sono impostati a '\0' (codice ASCII 0). Quando i dati vengono letti da questi campi, la stringa letta viene automaticamente completata con spazi (codice ASCII 32) fino alla sua dimensione massima. Se in un campo tipo **char** viene memorizzata una stringa vuota, nel campo viene scritto un singolo spazio per distinguerla da null.

Campo tipo char(10)	Valore dati
''	0x32000000000000000000
' '	0x32000000000000000000
'A '	0x41000000000000000000
'AaBb'	0x41614262000000000000
null	0x00000000000000000000

### ➤ Il tipo varchar

Questo tipo corrisponde al tipo SQL\_VARCHAR in ODBC. Viene utilizzato praticamente nello stesso modo del tipo **char**. Anche questo memorizza da 1 a 32511 caratteri.

Differisce dal tipo **char** perché quando i dati vengono inseriti in questo tipo di campo, non vengono cancellati gli spazi alla fine della stringa ma i byte corrispondenti nei file vengono riempiti con il carattere '\0'. Quando i dati vengono letti da questi campi, ODBC recupera la stringa senza modificarla. Come per il carattere tipo **char**, una stringa vuota viene memorizzata come singolo spazio per distinguerla da null.

Campo tipo varchar(10)	Valore dati
''	0x32000000000000000000
' '	0x32000000000000000000
'A '	0x41202020200000000000
'AaBb'	0x41614262000000000000
null	0x00000000000000000000



## ➤ Il tipo binary

Questo tipo corrisponde al tipo SQL\_BINARY in ODBC. Può memorizzare da 1 a 32511 caratteri.

Quando questo tipo di dati è utilizzato con CREATE TABLE, viene usato un byte extra (nonnull\_data). Questo byte può essere impostato a 0 oppure a 1 per distinguere il valore null da una stringa vuota. Se un tipo di campo **binary** viene utilizzato nella definizione di una chiave, il byte nonnull\_data viene incluso nella chiave con i byte di dati.

Nessun byte extra viene aggiunto quando il tipo binary è utilizzato con DEFINE TABLE. Vengono memorizzati soltanto i byte significativi. Il valore 0 non può quindi essere distinto dal valore null. Per questa ragione, questo tipo di campo non può essere null.

Quando vengono inseriti dati in questo tipo di campo, i byte corrispondenti nel file vengono completati con il carattere '\0'. In lettura di questo tipo di campo, tutti i byte sono ripristinati da ODBC.

La tabella che segue mostra i valori memorizzati. I dati in corsivo si applicano soltanto a CREATE TABLE.

Campo tipo binary(10)	<i>nonnull_data</i>	Valori dati
0x	<i>1</i>	0x00000000000000000000
0x00	<i>1</i>	0x00000000000000000000
0x1234	<i>1</i>	0x12340000000000000000
<i>null</i>	<i>0</i>	<i>0x00000000000000000000</i>

## ➤ Il tipo varbinary

Questo tipo corrisponde al tipo SQL\_VARBINARY in ODBC. Può memorizzare da 1 a 32511 caratteri.

Questo tipo può essere utilizzato soltanto con il comando CREATE TABLE. Nessun campo tipo **varbinary** può essere utilizzato con DEFINE TABLE. Per questo tipo di campo vengono usati due byte extra. I byte memorizzano la lunghezza dei dati binari più uno come intero corto. Il valore 0 corrisponde ad un null binario ed il valore 1 ad un binario di lunghezza zero.

Come il tipo **binary**, quando i dati vengono inseriti in questo tipo di campo, il blocco di file corrispondente viene completato con il carattere '\0'. In lettura di questo tipo di campo, soltanto i byte con lunghezza binaria vengono ripristinati da ODBC. Se un campo di tipo **varbinary** è utilizzato nella definizione di una chiave, i byte `size_data` non vengono inclusi con i byte dati nella chiave.

Campo tipo varbinary(10)	size_data	Valore dati
0x	0x0001	0x00000000000000000000
0x00	0x0002	0x00000000000000000000
0x1234	0x0003	0x12340000000000000000
null	0x0000	0x00000000000000000000

### ➤ Il tipo smallint

Questo tipo corrisponde al tipo `SQL_SMALLINT` in ODBC. Memorizza interi nell'intervallo da -32767 a 32767 in 2 byte. Il valore -32768 viene riservato per un campo null.

Campo tipo smallint	Valore dati
-32767	-32767
0	0
30000	30000
null	-32768

### ➤ Il tipo longint

Questo tipo corrisponde al tipo `SQL_INTEGER` in ODBC. Memorizza valori interi nell'intervallo da -134217727 a 134217727 in 4 byte. Il valore -134217728 viene riservato per un campo null.

Campo tipo longint	Valori dati
-32767	-32767
0	0
134217	134217
null	-134217728



### ➤ Il tipo real

Questo tipo corrisponde ai tipi SQL\_REAL e SQL\_FLOAT in ODBC. Memorizza numeri a virgola mobile in formato macchina in 4 byte.

Campo tipo real	Valori dati
-25	(float)-25.0
0	(float)0.0
3.1415	(float)3.1415
null	non-significant value

### ➤ Il tipo double

Questo tipo corrisponde al tipo SQL\_DOUBLE in ODBC. Memorizza numeri a virgola mobile in formato macchina in 8 byte.

Campo tipo double	Valori dati
-25	(double)-25.0
0	(double)0.0
3.1415	(double)3.1415
null	non-significant value

### ➤ Il tipo decimal

Questo tipo corrisponde al tipo SQL\_DECIMAL in ODBC. Memorizza numeri a virgola fissa.

Il tipo decimale deve essere seguito da due parametri tra parentesi, "c1 decimali(n, m)", dove n è il numero totale di cifre ed m il numero dei decimali. Se m non è specificato, viene utilizzato un valore predefinito di 0.

Le funzioni C-ISAM stdecimal() e lddecimal() sono utilizzate per leggere/scrivere campi di questo tipo.

### ➤ Il tipo data

Questo tipo corrisponde al tipo SQL\_DATE in ODBC. Memorizza una data come numero di giorni dal 1° gennaio dell'anno 0 come un intero di quattro byte. Per inserire o verificare una data con comandi SQL, utilizzare la notazione ODBC ({d 'AAAA-MM-JJ' }).

Campo tipo date	Valore dati
{ d '0000-01-01' }	0
{ d '1997-02-17' }	729438
null	-134217728

### ➤ Il tipo time

Questo tipo corrisponde al tipo SQL\_TIME in ODBC. Memorizza l'ora come numero di secondi nel giorno in un intero di 4 byte. Per inserire o verificare un'ora con comandi SQL, utilizzare la notazione ODBC. ({ t 'hh:mm:ss' }).

Campo tipo time	Valore dati
{ t '00:00:00' }	0
{ t '13:40:10' }	49210
null	-134217728

### ➤ Il tipo timestamp

Questo tipo corrisponde al tipo SQL\_TIMESTAMP in ODBC. Memorizza l'ora e la data come numero di secondi dal 1° gennaio 1970 (simile alla funzione time() in C). Il valore massimo corrisponde alla data del 5 febbraio 2036, ore 00.00. Per inserire o verificare la data e ora in comandi SQL, utilizzare la notazione ODBC ({ ts 'AAAA-MM-JJ hh:mm:ss' }).

Campo tipo timestamp	Valori dati
{ ts '1970-01-01 00:00:00' }	0
{ ts '1997-02-17 13:40:10' }	856186810
null	-134217728



# INDICE ANALITICO

---

## A

Allowed,88  
Ambiente,56  
Arrayfields (Progress),93  
Avvisi,78

## B

BackOffice,2  
binary,136, 139  
bit,136  
byte,136, 137

## C

Campi virtuali,57  
char,136, 138  
C-ISAM,37  
    ISAM-PATH,40  
    sqltools,38  
    SysColumns,38  
    SysDefaults,38  
    SysIndexes,38  
    SysTables,38  
Collegamenti tra tabelle,69  
COLUMN DEFINITION,106  
Comandi SQL/C-ISAM  
    COMMENT,114  
    CONNECT DATABASE,116  
    CREATE DATABASE,103  
    CREATE INDEX,112  
    CREATE SYNONYM,113  
    CREATE TABLE,104  
    DEFINE TABLE,105  
    DELETE,130  
    DISCONNECT DATABASE,117  
    DROP DATABASE,115  
    DROP INDEX,118  
    DROP SYNONYM,120  
    DROP TABLE,119, 121  
    INSERT,131  
    SELECT,122  
    UPDATE,133  
COMMENT,114  
config.XXX,86  
CONNECT DATABASE,116

CONSTRAINT DEFINITION,109, 110  
Convalida degli ambienti,74  
Create database,39  
CREATE DATABASE,103  
CREATE INDEX,112  
CREATE SYNONYM,113  
CREATE TABLE,104, 136

## D

Database virtuali,53  
date,136, 142  
DB2,14  
DBMAP.EXE,90  
DBSCRIPT.EXE,91  
DBSHOW.EXE,17, 92  
Debug,97  
decimal,136, 141  
DEFAULT,107  
DEFINE TABLE,105, 136  
DELETE,130  
Denied,88  
DISCONNECT DATABASE,117  
double,136, 141  
Driver ODBC virtuale,58  
DROP DATABASE,115  
DROP INDEX,118  
DROP SYNONYM,120  
DROP TABLE,119, 121

## E

Embedded SQL,9  
Esportazione di ambienti di sorgenti di dati,76  
EXPRESSION,124

## F

File  
    .dat,41  
    .dat files,37, 38, 40  
    .idx,41  
    .idx files,37, 38, 40  
    C-ISAM,37  
FILE IS,111  
FROM,125

	<b>G</b>	SET,134 VALUES,132 WHERE,126 Oracle,14, 96 ORDER BY,129 Origine dati,22
GROUP BY,127		
	<b>H</b>	
HAVING,128		
	<b>I</b>	param.XXX,93 Progress,14 Protezione,88
Importazione di una sorgente di dati,61		
Indice (C-ISAM),43		
Informix,14, 96		
INSERT,131		
ISAM-PATH,40		
	<b>J</b>	real,136, 141 Ridefinizione,56
Join,56		
	<b>L</b>	
Limiti,27		
longint,136, 140		
	<b>M</b>	Script (Esecuzione con sqltools),49 SELECT,122 SELECT CLAUSE,123 SET,134 smallint,136, 140 Sorgente dati virtuale,31 SQL,12 SQL_BINARY,136, 139 SQL_BIT,136 SQL_CHAR,136, 138 SQL_DATE,136, 142 SQL_DECIMAL,136, 141 SQL_DOUBLE,136, 141 SQL_FLOAT,136, 141 SQL_INTEGER,136, 140 SQL_REAL,141 SQL_SMALLINT,136, 140 SQL_TIME,136, 142 SQL_TIMESTAMP,136, 142 SQL_TINYINT,136, 137 SQL_VARBINARY,136, 139 SQL_VARCHAR,136, 138 sqltools,38 Sybase,14, 97 SysColumns,38 SysDefaults,38 SysIndexes,38 SysTables,38
Modello Client/Server,11		
	<b>N</b>	
NOT NULL,108		
	<b>O</b>	
ODBC,9		
Opzioni SQL/C-ISAM		
COLUMN DEFINITION,106		
CONSTRAINT DEFINITION,109, 110		
DEFAULT,107		
EXPRESSION,124		
FILE IS,111		
FROM,125		
GROUP BY,127		
HAVING,128		
NOT NULL,108		
ORDER BY,129		
SELECT CLAUSE,123		
	<b>P</b>	
	<b>R</b>	
	<b>S</b>	
	<b>T</b>	Tabelle (C-ISAM),41





Tabelle virtuali,56  
time,136, 142  
timestamp,136, 142  
Tipo  
  binary,136, 139  
  bit,136  
  byte,136, 137  
  char,136, 138  
  date,136, 142  
  decimal,136, 141  
  double,136, 141  
  longint,136, 140  
  real,136, 141  
  smallint,136, 140  
  time,136, 142  
  timestamp,136, 142  
  varbinary,136, 139  
  varchar,136, 138  
Tracciamento,96  
Traduttore,26

Tun SQL,13  
tunodbc200.XXX,95

## U

UPDATE,133

## V

VALUES,132  
varbinary,136, 139  
varchar,136, 138

## W

WHERE,126  
WOSA,9