

ESKER *Tun*[®] *Plus*

Tun SQL – Accès aux données

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.

PREAMBULE

Tun SQL-Accès aux données est un ensemble d'applications et de serveurs permettant à un PC de travailler en mode client/serveur sur des bases de données distantes (Informix, Oracle, Sybase, DB2, Progress, C-ISAM). **Tun SQL** utilise l'architecture ODBC définie par Microsoft.

Tun SQL fonctionne dans les environnements Windows version 3.x, Windows 95, Windows 98, Windows NT 3.51, Windows NT 4.0, Windows 2000 et dans les environnements Citrix WinFrame, Citrix MetaFrame et Windows NT TSE.

Tun SQL est intégré dans la gamme **Tun** de la manière suivante :

	Composants sous Windows	Composants en environnement multi-utilisateurs
Esker TCP/IP Stack	Couches de communication TCP/IP pour Windows 3.x seulement (DLL)	N/A
Accès aux ressources réseau	Applications TCP/IP (NIS, NFS client et serveur, PING, redirection et partage d'imprimantes, FTP client et serveur, Telnet VT320, RSH client et serveur, TAR, WALL, TFTP, TIME)	Applications TCP/IP (NFS client et serveur, PING, redirection et partage d'imprimantes, FTP client et serveur, Telnet VT320, RSH client, TAR, WALL)
Accès aux applications	Emulateur de terminal (émulations asynchrones, IBM3270, IBM5250, imprimantes 3287/3812)	Emulateur de terminal (émulations asynchrones, IBM3270, IBM5250, imprimantes 3287/3812)
Accès aux données	Drivers ODBC pour le mode Client/Serveur sur TCP/IP (SGBD Oracle, Informix, Sybase, DB2, Progress, C-ISAM), et outil de redéfinition de bases de données	Drivers ODBC pour le mode Client/Serveur sur TCP/IP (SGBD Oracle, Informix, Sybase, DB2, Progress, C-ISAM), et outil de redéfinition de bases de données
TCP/IP Network Services	Navigateur NIS, redirection et partage d'imprimantes	Redirection et partage d'imprimantes

La plupart des fonctionnalités et procédures décrites dans ce manuel peuvent être utilisées aussi bien sous Windows 3.x, Windows 95, Windows 98, Windows NT 3.51, Windows NT 4.0, Windows 2000 ou encore Citrix/Windows NT TSE. Cependant, certaines fonctionnalités ou procédures sont parfois spécifiques à un ou plusieurs de ces environnements. Dans ce cas, le paragraphe ou la section concernés sont balisés de la manière suivante :



Win 3.x

Spécificités Windows 3.x



Win 95

Spécificités Windows 95 et Windows 98



Win NT
2000

Spécificités Windows NT (Windows NT 3.51 et Windows NT 4.0, y compris environnement multi-utilisateurs si pas de précision) et Windows 2000



NT 4.0
2000

Spécificités Windows NT 4.0, y compris Citrix/Windows NT TSE si pas de précision



NT 3.51

Spécificités Windows NT 3.51, y compris environnement multi-utilisateurs



Win 32

Spécificités Windows 32 bits (Windows 95, Windows 98, Windows NT 3.51 et Windows NT 4.0, y compris environnement multi-utilisateurs si pas de précision, et Windows 2000)



Environnement multi-utilisateurs



Excepté en environnement multi-utilisateurs

Tun SQL pour Windows est systématiquement livré avec le produit **Tun PLUS** qui est le regroupement de tous les modules évoqués ci-dessus. La procédure d'installation du produit **Tun PLUS** proposera toujours l'installation de ce module. Cependant, excepté dans la version Multi-User Windows de **Tun PLUS**, **Tun SQL** peut s'installer indépendamment de **Tun PLUS**.

Remarque

Dans tout le document, les fonctionnalités sous Windows 98 sont identiques sous Windows 95.

SOMMAIRE

PARTIE 1 PRÉSENTATION ET UTILISATION

CHAPITRE 1 - Présentation de Tun SQL.....	1-11
Le mécanisme ODBC.....	1-11
Le modèle Client/Serveur.....	1-13
ODBC et le modèle Client/Serveur SQL.....	1-15
Le produit Tun SQL.....	1-16
CHAPITRE 2 - Configuration et utilisation sous Windows.....	2-21
Vérification du bon fonctionnement de Tun SQL.....	2-21
Création d'une base de données.....	2-25
Création d'une source de données.....	2-26
Téléchargement de la base de démonstration.....	2-33
Création d'une source de données virtuelle.....	2-35
Tables de conversion des caractères.....	2-38
CHAPITRE 3 - C-ISAM.....	3-41
Présentation de C-ISAM.....	3-41
Utilisation de sqltools.....	3-43

PARTIE 2 REVAMPING DE BASES DE DONNÉES

CHAPITRE 4 - Le revamping.....	4-57
Les bases de données virtuelles.....	4-57
Intégration du revamping dans Tun SQL.....	4-60
CHAPITRE 5 - Utilisation générale de Tun DB Revamp.....	5-63
Remarques générales.....	5-63
Importation des environnements d'une source de données.....	5-65
Création d'un environnement.....	5-66
Création d'une table virtuelle.....	5-67
Création d'un champ.....	5-68
Affectation de filtres à un champ.....	5-71
Liens entre tables réelles.....	5-73
Interrogation des bases de données réelle et virtuelle.....	5-75
Validation d'un environnement.....	5-78
Exportation des environnements d'une source de données.....	5-79
Mise à jour d'une source de données virtuelle.....	5-80
Création d'une source de données virtuelle.....	5-81
Affichage des avertissements.....	5-81
Gestion locale d'une source de données redéfinie.....	5-82

Identification des champs 5-83

PARTIE 3 ANNEXES

ANNEXE A - Références..... A-87

ANNEXE B - Ordres SQL pour C-ISAM..... B-105

Liste des principaux ordres B-105

Syntaxe des ordres SQL..... B-106

Types de données..... B-139

INDEX.....I-147

PARTIE 1
PRESENTATION ET UTILISATION

PRESENTATION DE TUN SQL

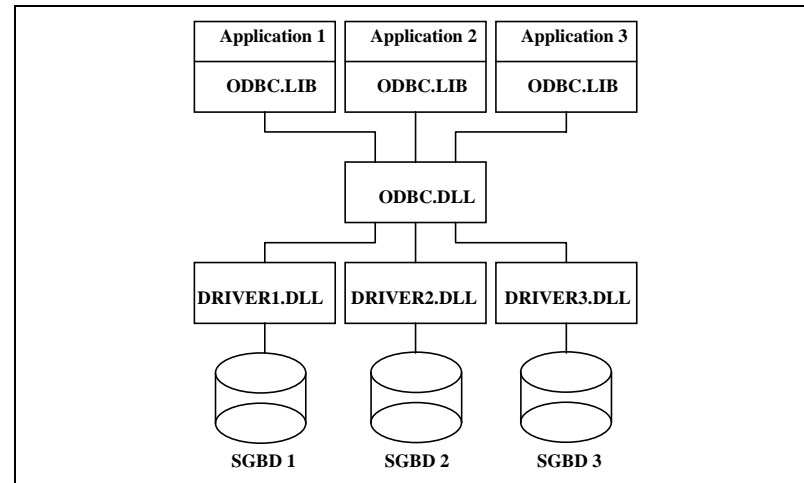
Le mécanisme ODBC

Dans le monde des bases de données, les programmeurs utilisent traditionnellement un mécanisme nommé "Embedded SQL" pour interfacier leurs applications avec une base de données spécifique. Le mécanisme "Embedded SQL" permet d'insérer des requêtes SQL au sein de programmes écrits en Cobol ou en C. Ce mécanisme offre l'avantage de pouvoir porter facilement les applications qui l'utilisent sur différentes machines.

Toutefois, le mécanisme "Embedded SQL" présente plusieurs inconvénients :

- Il existe autant d'"Embedded SQL" qu'il y a de moteurs de SGBD sur le marché. Les applications qui l'utilisent ne peuvent donc s'interfacier qu'avec un seul SGBD à la fois. Elles doivent être réécrites ou au moins modifiées pour pouvoir s'interfacier avec d'autres bases de données. Si les applications doivent s'interfacier avec toutes les bases de données du marché, il est impensable d'utiliser le mécanisme "Embedded SQL".
- Le mécanisme "Embedded SQL" est relativement pauvre, assez contraignant et difficile à mettre en oeuvre. Il ne permet pas de tirer partie de toutes les possibilités offertes par une base de données ; il est parfois préférable d'utiliser directement l'API offerte par le SGBD.

Pour pallier les inconvénients du mécanisme "Embedded SQL", la société Microsoft a imaginé une nouvelle approche au travers du mécanisme ODBC (Open DataBase Connectivity) qui s'inscrit dans le cadre de l'architecture WOSA (Windows Open System Architecture).



ODBC est un jeu de fonctions C parfaitement définies qui permettent d'extraire ou de mettre à jour des données issues d'un SGBD. Ces fonctions sont regroupées au sein d'une DLL (Dynamic Link Library) qui peut être utilisée par n'importe quelle application Windows.

Les fonctions de la DLL ODBC (ODBC.DLL) analysent les requêtes SQL et les sous-traitent auprès de drivers (drivers ODBC) qui sont chargés de convertir les appels dans l'API spécifique du SGBD que l'on souhaite utiliser. Un driver ODBC permet de virtualiser l'interface d'un SGBD et de faire en sorte qu'une application puisse l'utiliser comme n'importe quel autre SGBD compatible ODBC.

La société Microsoft fournit la bibliothèque ODBC.DLL et tous les outils qui permettent de l'utiliser. Elle ne fournit pas, en revanche, les drivers ODBC pour chacun des SGBD du marché. Elle se contente de fournir les drivers pour ses propres systèmes de stockage (**Access, Excel, Word,...**).

Les drivers ODBC spécifiques à chacune des bases de données peuvent être directement fournis par l'éditeur du SGBD ou par des sociétés tierces qui en ont fait leur spécialité (Esker).



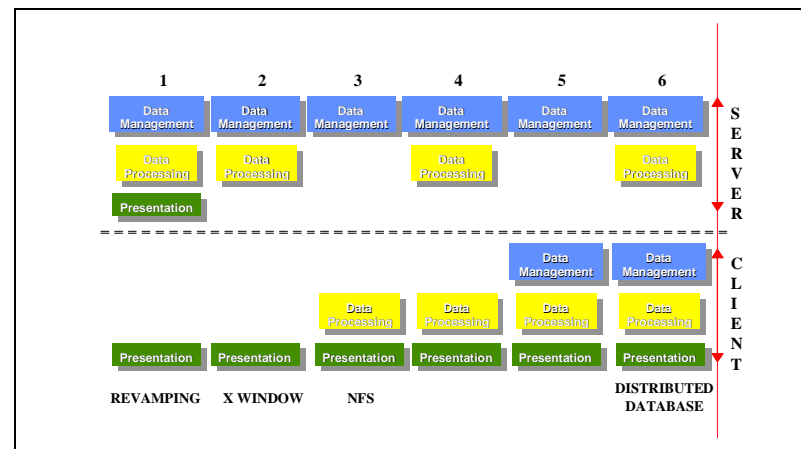
En conclusion, le mécanisme ODBC offre une interopérabilité maximale. Une simple application Windows peut accéder à différents systèmes de gestion de données sans avoir été au préalable conçue pour cela. ODBC permet aux développeurs de programmer, de compiler et de livrer leurs programmes sans avoir à se soucier du SGBD sur lequel l'application sera exploitée. Les utilisateurs n'ont qu'à fournir le driver ODBC adéquat pour que l'application qui leur est fournie puisse tourner sur le SGBD de leur choix.

ODBC est un mécanisme précieux pour les applications horizontales telles que les tableurs, les traitements de texte et les outils de développement qui peuvent ainsi exploiter des informations issues de n'importe quel SGBD sans connaître a priori celui qui sera utilisé.

Le modèle Client/Serveur

Depuis quelques années, l'industrie informatique ne parle plus que du mode Client/Serveur. Dans sa définition la plus large, le modèle Client/Serveur est une conception de l'informatique dans laquelle deux unités centrales au moins coopèrent pour fournir un service particulier.

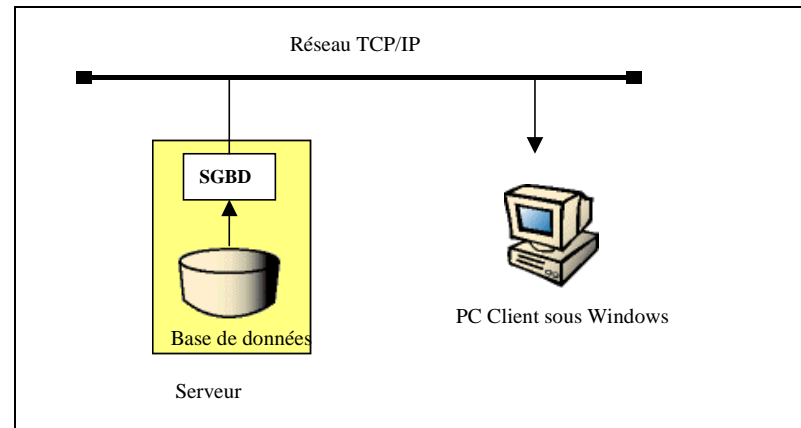
Le "Gartner Group" a recensé six grands types d'application du mode Client/Serveur qu'il a répartis en fonction du nombre de fonctions réalisées par le client ou le serveur :



Les applications qui sollicitent le moins le "client" sont les applications de "revamping" ou les serveurs X-WINDOW. Les applications qui sollicitent le plus le "client" sont les applications qui utilisent des bases de données réparties.

Lorsque l'industrie informatique parle du modèle Client/Serveur, c'est très souvent aux applications sur bases de données réparties auxquelles elle fait référence.

L'illustration la plus populaire de ce mode est la suivante :



Un PC "client" dispose d'une application de gestion traditionnelle en mode graphique. Les données sont centralisées sur un serveur UNIX et sont gérées par un Système de Gestion de Base de Données Relationnel (SGBDR). Pour extraire ou pour mettre à jour des données, le client transmet une requête SQL au serveur qui l'exécute et retourne la réponse au client par l'intermédiaire du réseau.

Les principaux avantages de cette architecture sont les suivants :

- L'utilisateur de l'application dispose d'une interface homme/machine graphique et conviviale sur son PC au travers du système d'exploitation Windows.
- Le PC peut servir à autre chose qu'à utiliser l'application considérée (bureautique, calcul, applications personnelles...).
- Alors qu'il dispose d'une machine personnelle, l'utilisateur peut avoir accès à des données centralisées sur un serveur en même temps que d'autres utilisateurs.



- Le serveur est débarrassé de la partie applicative et toute sa puissance est consacrée à servir des données. La répartition de la charge de travail est équilibrée entre le client et le serveur.
- Le réseau ne sert à véhiculer que les données essentielles de l'application, il n'est pas surchargé par les informations de présentation.

Le mode Client/Serveur SQL tel qu'il est décrit ci-dessus est une version modernisée du mode transactionnel tel qu'on le rencontre encore autour des Main Frame en coopération avec des terminaux synchrones.

ODBC et le modèle Client/Serveur SQL

Dans la mesure où le mécanisme ODBC banalise l'accès à tous les systèmes de gestion de données, il peut être aussi utilisé si la base de données est distante. Dans ce contexte, ODBC permet à une application Windows d'exploiter n'importe quel SGBD distant, quelle que soit son origine. Il permet aussi à des applications verticales telles que **Excel**, **Word** ou **Access** d'exploiter les informations centralisées de l'entreprise. Dans un contexte Client/Serveur, le mécanisme ODBC permet de multiplier les applications qui peuvent utiliser les données de l'entreprise.

Toutefois, la mise en oeuvre d'une telle architecture aujourd'hui n'est pas une sinécure. En effet, un utilisateur disposant déjà d'un réseau local de PC, d'un serveur UNIX et d'un SGBD, doit acquérir en outre les composants suivants pour pouvoir mettre en oeuvre une architecture Client/Serveur SQL sur ODBC :

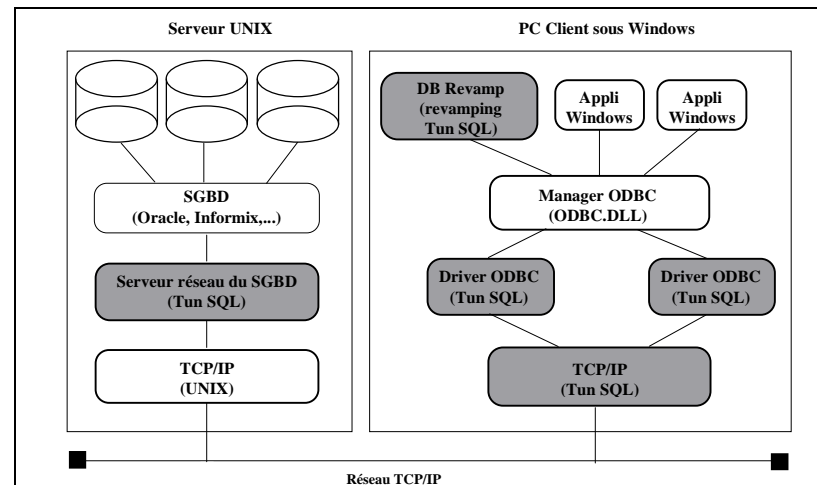
- La partie serveur réseau du SGBD (Informix Net, Sql Net...).
- La partie cliente PC du SGBD (Informix Net PC, Sql Net PC...).
- Un driver ODBC approprié.
- Des couches de communication TCP/IP compatibles 'Winsock' pour PC.

Les éditeurs de SGBD fournissent toujours les deux premiers composants. Ce n'est pas toujours le cas pour le driver ODBC et ce n'est jamais le cas pour les couches de communication TCP/IP. Il convient donc de se procurer par ailleurs ces deux derniers composants en faisant bien attention aux problèmes de compatibilité des composants entre eux.

Le produit Tun SQL

Tun SQL a été conçu pour résoudre définitivement ce problème. Il intègre en un seul produit homogène l'ensemble des composants évoqués ci-dessus, ainsi qu'une fonctionnalité de revamping de bases de données et un driver ODBC virtuel permettant d'accéder aux bases de données redéfinies.

Même les composants réseau du SGBD (client et serveur) font partie de **Tun SQL**. Cela représente une économie substantielle lorsqu'il s'agit d'équiper de nombreux postes PC. Cela apporte surtout une grande simplicité lorsqu'il s'agit de mettre en oeuvre une architecture Client/Serveur SQL.



Les principales caractéristiques du produit **Tun SQL** sont décrites dans les paragraphes suivants.



➤ Un driver ODBC unique pour la majorité des SGBD

Afin de limiter le nombre de produits à installer sur le PC client, **Tun SQL** intègre un unique driver ODBC pour accéder indifféremment aux SGBD suivants :

- Oracle version 7.
- Informix version 5 et version 7.
- Sybase version 10.
- DB2 version 2.
- Progress versions 6, 7 et 8.
- C-ISAM versions 4 à 7.

➤ Une application de "revamping" de bases de données

Grâce à une application conviviale, **Tun SQL** offre la possibilité de redéfinir les tables d'une base de données, afin de les rendre plus accessibles aux utilisateurs finals (réorganisation des tables en fonction des besoins, modification des noms des tables et champs, pré-calculs).

➤ Un driver ODBC virtuel permettant d'accéder aux bases de données redéfinies par revamping

Afin de pouvoir utiliser une base de données redéfinie par revamping, **Tun SQL** intègre un driver ODBC virtuel chargé de traduire les requêtes effectuées sur des tables virtuelles, en requêtes compréhensibles par un driver ODBC classique.

➤ La partie serveur des SGBD fait partie de Tun SQL

La partie serveur de chacun de ces SGBD s'installe sous UNIX ou NT et est fournie en standard avec le produit **Tun SQL** pour les systèmes d'exploitation suivants :

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

- Windows NT
- IBM MVS

Cette caractéristique permet aux utilisateurs de **Tun SQL** d'économiser l'achat de la partie serveur des SGBD qu'ils utilisent.

➤ Des couches de communication TCP/IP en standard sous Windows 16 bits



Win 3.x

Comme tous les produits de la gamme **Tun**, **Tun SQL** est livré en standard avec les couches de communication TCP/IP de la société Esker sous Windows 16 bits (**Esker TCP/IP Stack**). Ces couches de communication offrent un excellent niveau de performance et ont été validées avec tous les composants de **Tun SQL**. Leur présence dans le produit **Tun SQL** évite à l'utilisateur de devoir se les procurer par ailleurs.

➤ Une simplicité d'installation et d'administration

La raison d'être de **Tun SQL** est de faciliter la mise en oeuvre d'une architecture Client/Serveur entre Windows et UNIX. C'est la raison pour laquelle **Tun SQL** offre une procédure d'installation simple pour UNIX et NT et pour Windows ainsi qu'une documentation complète.

En plus du driver ODBC, deux applications sont livrées sous Windows qui permettent de tester et d'utiliser la configuration Client/Serveur :

- **Tun DB Show** qui permet de tester la liaison Client/Serveur de bout en bout. Cette application permet d'interroger un serveur UNIX et NT pour connaître les SGBD qui sont installés et d'interroger certains SGBD pour connaître les bases de données qu'ils contiennent.
- **Tun DB Script** qui permet d'exécuter sous Windows des scripts SQL afin de créer des bases de données sur un SGBD distant.

Outre la sécurité offerte par UNIX et les différents SGBD, **Tun SQL** fournit un mécanisme qui permet d'interdire à certaines applications Windows l'accès à certaines bases de données sensibles.





Enfin, l'intégration dans **Tun SQL** du mécanisme NIS (Network Information System) permet la gestion centralisée des ressources du réseau et facilite l'accès aux ressources distantes. Pour plus de précision sur NIS, consultez le manuel **TCP/IP Network Services** ou le manuel **Tun NET**.

➤ Niveau de conformité du driver Tun SQL

Le driver **Tun SQL** supporte toutes les fonctions du niveau 1. Il supporte quelques fonctions du niveau 2. De plus, la bibliothèque "Microsoft ODBC Cursor Library" est fournie avec le driver. Bien que cette bibliothèque ne supporte que les curseurs statiques et "forward only", elle satisfait les besoins de beaucoup d'applications.

CONFIGURATION ET UTILISATION SOUS WINDOWS

Vérification du bon fonctionnement de Tun SQL

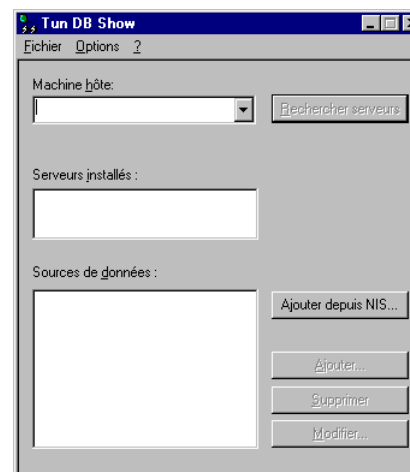
➤ Lancement de Tun DB Show

Après l'installation et la configuration de **Tun SQL** sous Windows et sous UNIX, il est nécessaire de vérifier son bon fonctionnement. Pour ce faire, il suffit de lancer l'application **Tun DB Show**.



Lancez le programme en cliquant sur l'icône **Tun DB Show** du groupe **Data Access** (menu **Démarrer**, **Programmes**, **Esker Tun** sous Windows 95/98/2000 et Windows NT).

La fenêtre suivante apparaît :



Cet utilitaire permet d'interroger un serveur du réseau pour vérifier si un ou plusieurs serveurs de **Tun SQL** y sont installés.

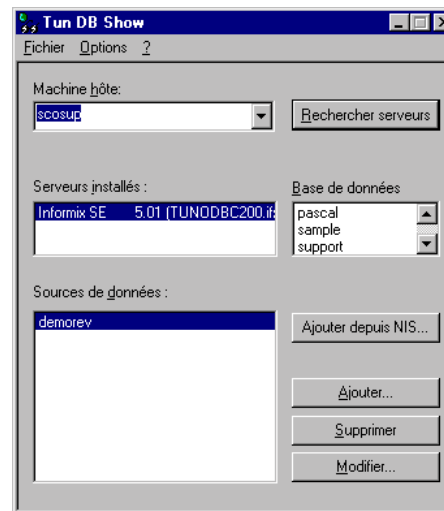
Entrez le nom ou l'adresse IP du serveur dans le champ **Machine hôte**, ou sélectionnez le serveur de votre choix dans la liste déroulante (cette liste propose les serveurs déclarés dans le fichier **hosts** et sur le serveur NIS).



Pour la configuration de **Tun NIS**, consultez le manuel **TCP/IP Network Services** ou le manuel **Tun NET**.

Cliquez ensuite sur le bouton **Rechercher serveurs**.

Si un ou plusieurs serveurs de **Tun SQL** sont convenablement installés sur la machine distante, alors ceux-ci apparaissent dans la liste **Serveurs installés** (au moins une ligne) comme sur la figure suivante :



Chaque ligne affiche les informations suivantes :

- Le nom du SGBD avec lequel est interfacé le serveur de **Tun SQL**.
- Le numéro de version du SGBD.
- Le nom de l'exécutable assurant la fonction de serveur (**tunodbc200.ora** par exemple).



Remarque

Pour certains SGBD (Informix On Line par exemple), le fait de sélectionner le serveur dans la liste a pour effet d'afficher dans une autre liste l'ensemble des bases de données qui sont gérées par le SGBD.

Si aucun serveur de **Tun SQL** n'apparaît dans la liste, c'est qu'il y a eu un problème lors de l'installation. Il convient alors de refaire toutes les opérations et toutes les vérifications décrites dans le guide d'installation.

➤ Paramétrage

L'application **Tun DB Show** dispose d'un menu **Options** par lequel il est possible notamment de :

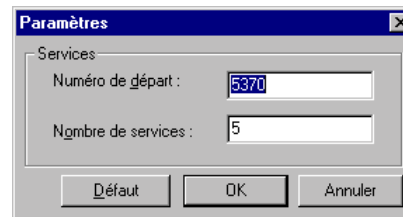
- Gérer les services.
- Paramétrer l'utilisation d'un serveur Proxy.

Gestion des services

A chaque processus serveur de **Tun SQL** est associé un numéro de service numéroté à partir de 5370. Il est nécessaire pour le bon fonctionnement de **Tun SQL** que soient définis le numéro de départ des services et le nombre de services possibles, afin de pouvoir détecter les différents systèmes de bases de données disponibles sur un serveur **Tun SQL**. Par défaut, le numéro de départ est 5370 et le nombre de services à parcourir est 5. Ces valeurs peuvent être modifiées. La liste des services possibles est la suivante :

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

Sélectionnez l'option **Options**→**Paramètres...** du menu général. La boîte de dialogue suivante apparaît :

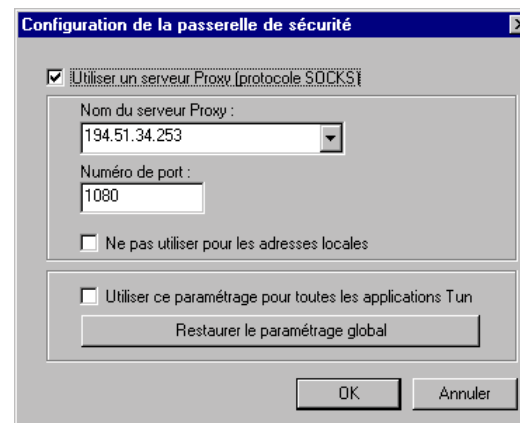


Utilisation d'une passerelle de sécurité de type Proxy

Tout accès à un ordinateur situé hors du réseau local peut être soumis au passage par une passerelle de sécurité de type Proxy.

Pour fixer les paramètres de la passerelle de sécurité (adresse IP, numéro de port,...), sélectionnez l'option **Options**→**Passerelle de sécurité** du menu général.

La boîte de dialogue suivante apparaît :



Pour procéder à la configuration de la passerelle de sécurité, sélectionnez la case à cocher **Utiliser un serveur Proxy (protocole SOCKS)**. Entrez le nom ou l'adresse IP du serveur (n'entrez un nom que si vous disposez d'un DNS). Utilisez éventuellement la liste des valeurs proposées : elle correspond aux serveurs enregistrés dans la table des serveurs (**hosttab**) ainsi que sur le serveur NIS (les ressources NIS sont représentées en jaune).

Entrez également le port correspondant au protocole SOCKS (en général 1080, valeur proposée par défaut).

Si vous voulez éviter le passage par la passerelle de sécurité pour toute connexion à une adresse locale, sélectionnez la case à cocher **Ne pas utiliser pour les adresses locales**.

Enfin, la configuration de la passerelle de sécurité peut être étendue à toutes les applications **Tun** en sélectionnant la case à cocher **Utiliser ce paramétrage pour toutes les applications Tun**. Pour appliquer la configuration générale à toutes les applications **Tun** en cours (par exemple, après avoir utilisé une configuration spécifique à **Tun DB Show**), cliquez sur le bouton **Restaurer le paramétrage global**.

Création d'une base de données

Le produit **Tun SQL** est livré avec des exemples d'utilisation afin de familiariser l'utilisateur avec le mécanisme ODBC en environnement client/serveur.

Pour pouvoir exploiter les exemples fournis avec le produit **Tun SQL**, il est nécessaire de créer une base de données spécifique sur l'un ou l'autre des SGBD disponibles sur votre système. Pour ce faire, il suffit d'utiliser les outils propres à chacun des SGBD. Compte tenu de la difficulté à créer une base de données sur certains SGBD (Oracle), il est possible d'utiliser une base existante, pourvu que les données qu'elle contient ne soient pas "sensibles".

Pour la bonne compréhension de la suite de la documentation, il est préférable que la nouvelle base de données créée s'appelle **tunsqldemo**.

Création d'une source de données

➤ Principe des sources de données

Pour pouvoir exploiter tel driver plutôt qu'un autre et telle base de données plutôt qu'une autre, les applications compatibles ODBC ont besoin de connaître une "**source de données**". Une "**source de données**" est un mot-clé sous lequel sont regroupés le nom du driver ODBC utilisé (**tunodb32.dll** par exemple) et les informations nécessaires au fonctionnement de ce driver. Les informations dont a besoin le driver ODBC sont les suivantes :

- Le nom ou l'adresse IP du serveur distant.
- Le type du SGBD (Oracle, Informix, Sybase, DB2, Progress, C-ISAM).
- Le nom de la base de données.
- Un commentaire.
- Des informations complémentaires facultatives.

➤ Création d'une source de données

Les exemples fournis avec le produit **Tun SQL** exploitent la même source de données (sauf les exemples concernant le revamping de bases de données). Afin de pouvoir suivre ces exemples (voir "**Bien démarrer avec Tun**"), vous devez créer cette source de données.

Pour créer une source de données, vous pouvez utiliser :

- **Tun DB Show**.
- **ODBC DataSource Administrator**, utilitaire Windows.

Création d'une source de données depuis Tun DB Show

Lancez de nouveau l'application **Tun DB Show** et réalisez les opérations suivantes :

- Entrez le nom ou l'adresse IP du serveur sur lequel la base de données pour laquelle vous voulez créer une source de données, a été installée.
- Cliquez sur le bouton **Rechercher serveurs** pour faire apparaître la liste des serveurs de **Tun SQL**.



- Sélectionnez le serveur associé au SGBD contenant la base de données pour laquelle vous voulez créer une source de données.
- Cliquez sur le bouton **Ajouter**.



Si l'application **Tun NIS** a été installée sur le PC et si l'administrateur du réseau a procédé à la configuration des tables NIS, le bouton **Ajouter depuis NIS...** permet d'accéder à la liste des sources de données disponibles sur le réseau. Pour la configuration de **Tun NIS**, consultez le manuel **TCP/IP Network Services** ou le manuel **Tun NET**.

Passez ensuite au paragraphe "**Configuration de la source de données**".

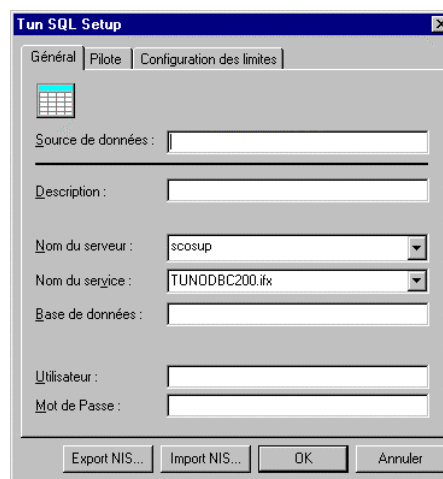
Création d'une source de données depuis ODBC Administrator

Dans le **Panneau de Configuration**, cliquez sur l'icône **ODBC** (32bit ODBC sous Windows 32 bits). Dans la boîte de dialogue qui apparaît, cliquez sur le bouton **Ajouter** en version française ou **Add** en version anglaise.

Sélectionnez ensuite le pilote ODBC **Tun32 Driver**.

Configuration de la source de données

Cliquez sur le bouton **Ajouter** depuis **Tun DB Show** ou depuis l'**ODBC Administrator**. La boîte à onglets suivante apparaît :



➤ Onglet Général

Source de données



Cette icône représente le champ contenant le nom de la source de données telle qu'elle sera utilisée par les applications compatibles ODBC. Pour atteindre la base de données exemple, il est impératif que la source de données s'appelle :

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

Description

Ce champ est un commentaire associé à la source de données.

Nom du serveur

Ce champ doit contenir l'adresse IP ou le nom du serveur sur lequel a été installée la base de données que l'on souhaite utiliser.

Nom du service

Ce champ doit contenir le nom du processus serveur de **Tun SQL** associé au SGBD sur lequel la base de données que l'on souhaite exploiter a été créée (**tunodbc200.ora** par exemple).

Si vous utilisez d'autres couches TCP/IP que celles fournies dans le produit **TCP/IP Stack**, vous devez renseigner le fichier **services** du logiciel TCP/IP utilisé par les valeurs suivantes :

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



Base de données

Ce champ doit contenir le nom de la base de données que l'on souhaite exploiter. Pour pouvoir exploiter la base de données exemple, entrez **tunsqldemo** qui est le nom de la base de données fournie par **Tun SQL**.

Utilisateur

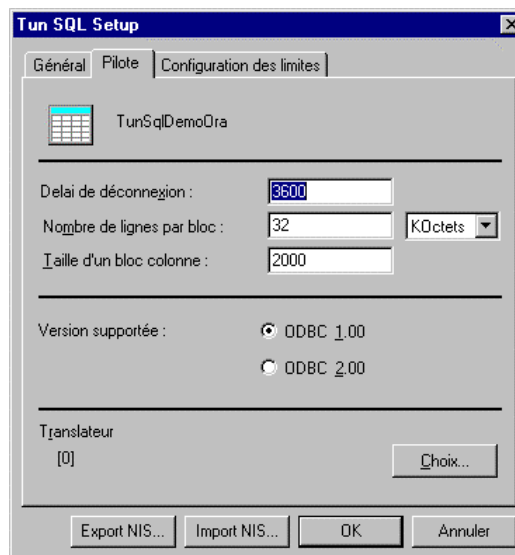
Ce champ doit contenir un nom d'utilisateur autorisé à accéder à la base de données.

Mot de passe

Ce champ doit contenir le mot de passe associé à l'utilisateur.

➤ Onglet Pilote

Cliquez sur l'onglet **Pilote** pour faire apparaître la fenêtre de configuration du pilote ODBC :



Délai de déconnexion

Le PC étant une machine soumise à de fréquentes interruptions logicielles ou matérielles, il est nécessaire que le serveur de **Tun SQL** vérifie régulièrement si le PC est encore sous tension. Pour effectuer cette vérification, il envoie régulièrement des paquets en direction du PC. Si celui-ci ne répond pas dans un délai de "n" secondes alors le processus s'arrête. Ce champ est prévu pour renseigner la valeur du timeout ou délai de déconnexion (1 heure par défaut).

Nombre de lignes par bloc

Ce champ permet d'indiquer la taille des paquets de données extraits d'une table lors d'un "select" SQL. La valeur peut être exprimée en Koctets ou en nombre de lignes.

Si cette valeur est égale à "1 ligne" alors il y aura une rangée extraite par paquet TCP. Si cette valeur est égale à "100" alors les rangées seront regroupées par paquet de 100 à concurrence du nombre de rangées réellement extraites. Cette valeur permet d'optimiser les échanges sur le réseau. La valeur optimale est comprise entre 50 et 150. La valeur par défaut en Koctets est 32.

Taille d'un bloc colonne

Ce champ permet d'indiquer l'unité de fragmentation lorsqu'il s'agit de récupérer des colonnes très larges dans la base de données (grands textes ou images). Si cette valeur est trop faible, cela aura pour effet d'augmenter considérablement les échanges sur le réseau.

Version supportée

Il existe à ce jour 2 versions de l'API ODBC ; la version 1.00 et la version 2.00. Certaines applications ne sont compatibles qu'avec la version 1.00 d'ODBC (**Access** de Microsoft) et ne fonctionnent pas avec des drivers d'une version supérieure. Pour que ces applications puissent tout de même s'exécuter sur le driver ODBC de **Tun SQL** qui est compatible avec la version 2.00, celui-ci est capable d'émuler une version 1.00.

Selon le niveau de compatibilité de l'application qui utilisera la source de données vous pouvez indiquer au driver ODBC le niveau de compatibilité qu'il doit offrir en sélectionnant la case à cocher appropriée.



Translateur

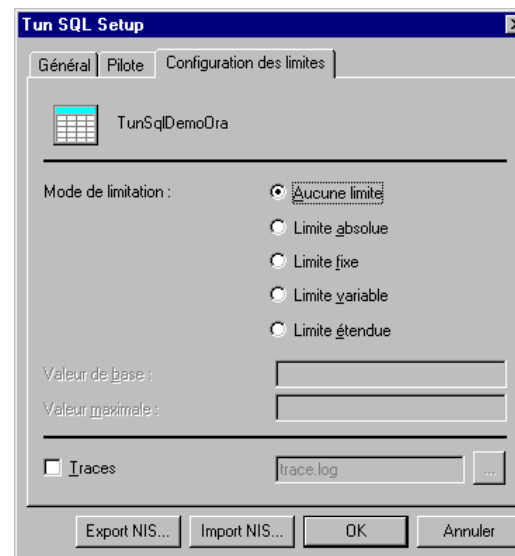
Compte tenu de la différence de notation des caractères accentués entre les environnements Windows (CP850) et UNIX (ISO8859), il est parfois nécessaire de mettre en place des tables de conversion de caractères au niveau du driver ODBC. Le bouton **Choix...** permet de choisir la table de conversion nécessaire. Dans le cas de la base de données exemple, il n'est pas nécessaire de renseigner ce champ car tous les textes qu'elle contient sont libellés en anglais.

Remarque :

Les tables de conversions peuvent être créées ou modifiées à l'aide de l'application **Tun DB Map**.

➤ Onglet Configuration des limites

Cliquez sur l'onglet **Configuration des limites** pour faire apparaître la fenêtre de configuration des limites :



Mode de limitation

Certaines applications bureautiques laissent l'utilisateur libre de composer ses propres requêtes SQL. Par ailleurs, certaines applications ont tendance à lire le contenu de toute une table avant de l'afficher à l'écran. Cela ne pose pas de problèmes particuliers lorsqu'il s'agit de petites tables situées sur une base de données locale. En revanche, cela pose des problèmes insurmontables lorsqu'il s'agit de très grandes tables situées sur une base de données centralisée et distante.

Dans ce dernier cas, les échanges sur le réseau sont considérables et la mémoire du PC devient insuffisante pour stocker les données reçues. Très souvent le PC est obligé de "rebooter" après une telle requête "select".

Le driver ODBC de **Tun SQL** reconnaît cinq types de limites :

Aucune limite	Aucune limite n'est imposée par le driver ODBC
Limite absolue	Le driver ODBC refusera de charger plus de "n" rangées en une seule requête "select". Il n'affichera pas de messages pour en informer l'utilisateur.
Limite fixe	Le driver ODBC refusera de charger plus de "n" rangées en une seule requête "select". Il affichera toutefois un message pour en informer l'utilisateur.
Limite Variable	Le driver ODBC refusera de charger plus de "n" rangées en une seule requête "select". Il affichera toutefois un message proposant d'en charger davantage dans la limite de la valeur supérieure (Valeur maximale)
Limite étendue	Le driver ODBC refusera de charger plus de "n" rangées en une seule requête "select". Il affichera toutefois un message proposant d'en charger davantage sans limite supérieure. Dans ce dernier cas, le message affiché n'est qu'un "Warning".

Traces

En sélectionnant la case à cocher **Traces**, vous pouvez choisir d'enregistrer la trace de vos requêtes SQL dans un fichier ".log" que vous pourrez ensuite consulter. Cette option vous permet de comprendre ce que le driver ODBC effectue lorsque vous lui soumettez une requête SQL.



Pour choisir le répertoire dans lequel vous souhaitez placer ce fichier, ainsi que son nom, cliquez sur le bouton "...".

NIS



Si l'application **Tun NIS** a été installée sur le PC et si l'administrateur du réseau a procédé à la configuration des tables NIS, le bouton **Import NIS...** permet d'accéder à la liste des sources de données disponibles sur le réseau. Pour la configuration de **Tun NIS**, consultez le manuel **TCP/IP Network Services** ou le manuel **Tun NET**.

Remarques

Dans le cadre de ce chapitre, nous vous avons demandé de créer une source donnée ayant pour nom **tunsqldemoXXX** car c'est à cette source de données que se réfère l'ensemble des exemples fournis avec le produit **Tun SQL**.

Si vous souhaitez utiliser **Tun SQL** avec d'autres applications et d'autres bases de données, il faudra recréer à chaque fois la source de données adéquate. En règle générale, il doit exister une source de données spécifique par application exploitée et par base utilisée.

Téléchargement de la base de démonstration

Afin de pouvoir utiliser les exemples fournis avec le produit **Tun SQL**, celui-ci est livré avec une base de données exemple. Cette dernière doit être téléchargée à partir du PC sur la source de données **tunsqldemoXXX** que nous vous avons demandé de créer dans les sections précédentes.

Remarque

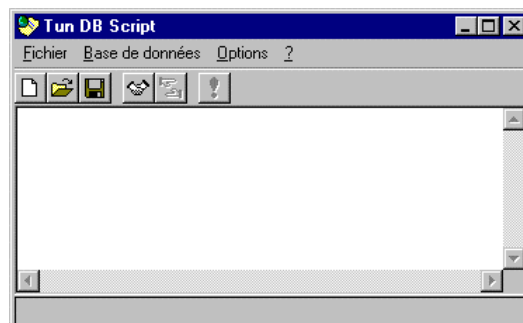
Dans le nom de la source de données, la variable "XXX" peut prendre l'une des valeurs suivantes :

- ifx pour Informix On Line
- ise pour Informix SE
- ora pour Oracle
- syb pour Sybase




Pour réaliser ce téléchargement, lancez le programme en cliquant sur l'icône **Tun DB Script** du groupe **Data Access** (menu **Démarrer**, **Programmes**, **Esker Tun** sous Windows 95/98/2000 et Windows NT).


La fenêtre suivante apparaît :



➤ **Chargement du script SQL de création de la base de données**

Sélectionnez l'option **Fichier**→**Ouvrir...** ou cliquez sur le bouton  pour charger le script SQL de votre choix. Pour télécharger la base exemple, il est nécessaire de charger le fichier **Demo\Db\Xxxcreat.sql** à partir du répertoire d'installation du produit **Tun SQL**.


➤ **Connexion sur la source de données**

Avant de pouvoir exécuter le script SQL, il est nécessaire de se connecter sur une source de données. Pour ce faire, cliquez sur le bouton  ou utilisez l'option **Base de données**→**Connecter**. Cette opération va provoquer l'affichage d'une boîte de dialogue qui demandera le nom de la source de données à utiliser. Si la connexion peut s'établir, l'exécution du batch peut avoir lieu.


Pour exécuter le téléchargement de la base de données exemple, sélectionnez la source de données **TunSqlDemoXXX** définie précédemment.



➤ Exécution

Pour exécuter le script SQL, sélectionnez l'option **Base de données** ➔ **Exécuter** du menu général ou cliquez sur le bouton . L'application **Tun DB Script** soumettra alors les ordres SQL les uns après les autres à la base de données correspondant à la source de données sélectionnée. En cas d'erreur, l'application s'arrêtera et affichera un message d'erreur.

➤ Déconnexion de la source de données

Après exécution, il est nécessaire de se déconnecter de la source de données en cliquant sur le bouton  ou en sélectionnant l'option **Base de données** ➔ **Déconnecter** du menu général.

Le fait de quitter l'application provoque aussi une déconnexion de la source de données.

Remarque

Si la première utilisation de l'application **Tun DB Script** est de télécharger la base exemple du produit **Tun SQL**, il est également possible d'utiliser **Tun DB Script** pour d'autres usages. **Tun DB Script** permet en effet d'exécuter des trains entiers d'ordres SQL pour créer d'autres bases de données ou pour mettre à jour ou purger certaines tables de façon massive.

Création d'une source de données virtuelle

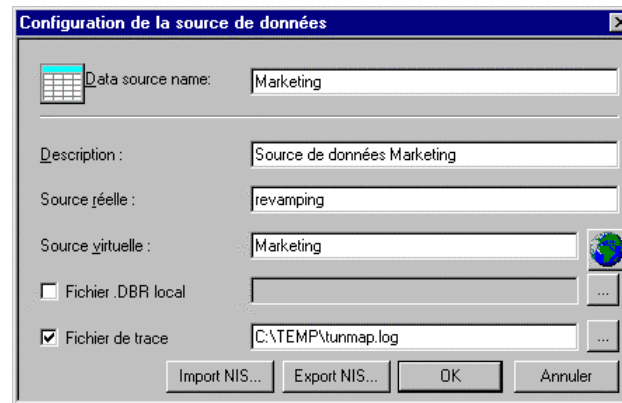
L'application **Tun DB Revamp** permet d'associer à une base de données réelle en ensemble de tables virtuelles, adaptées au contexte des utilisateurs finals. Voir la partie "**Revamping de bases de données**" pour en savoir plus sur cette application.

Lorsqu'une base de données virtuelle (ou "revampée") a été créée, vous pouvez comme pour une base de données réelle, créer des sources de données relatives à cette base. Elles permettront aux utilisateurs d'exploiter la base de données virtuelle spécifiquement créée pour eux, en utilisant le driver ODBC virtuel fourni par **Tun SQL**.

Une source de données virtuelle peut être vue comme l'association d'une source de données réelle et d'un environnement.

Pour créer une source de données virtuelle, accédez à la configuration de la source de données virtuelle depuis l'**ODBC Administrator** en choisissant le driver **Tunmap32**. Voir "**Création d'une source de données**".

La boîte à onglets suivante apparaît :



Source de données

Entrez le nom de la source de données revampée.

Description


Complétez ce champ comme vous le souhaitez, afin de décrire la source de données créée.

Source réelle

Entrez le nom de la source de données correspondant à la base de données réelle dont est issue la base de données virtuelle.




Source virtuelle

Entrez le nom de l'environnement pour lequel vous créez une source de données virtuelle. Un environnement est le regroupement de tables virtuelles, chaque base de données ayant subi un "revamping" peut disposer d'un ou plusieurs environnements. Cliquez sur le bouton  pour choisir l'environnement dans la liste des environnements définis dans la base de données.


Fichier .DBR local

Au lieu d'entrer le nom d'une source de données réelle, vous pouvez choisir de sélectionner l'environnement depuis un fichier ".dbr" local. Pour comprendre l'utilité et l'utilisation d'un fichier de ce type, reportez-vous à la partie "**Revamping de bases de données**".

Sélectionnez dans ce cas la case à cocher **Fichier .DBR local**. Entrez le nom du fichier, chemin d'accès compris, ou cliquez sur le bouton  pour sélectionner ce fichier. Dans ce cas, choisissez ensuite l'environnement (champ **Source virtuelle**).

Traces

En sélectionnant la case à cocher **Fichier de trace**, vous pouvez choisir d'enregistrer la trace de vos requêtes SQL dans un fichier ".log" que vous pourrez ensuite consulter. Cette option vous permet de comprendre ce que le driver ODBC effectue lorsque vous lui soumettez une requête SQL.

Pour choisir le répertoire dans lequel vous souhaitez placer ce fichier, ainsi que son nom, cliquez sur le bouton  .

Tables de conversion des caractères

Ce chapitre peut être évité en première lecture.

➤ Différence de notation entre les différents systèmes informatiques

Si tous les caractères utilisés par la langue anglaise ont été parfaitement codifiés par la table ASCII (de 0 à 127), il n'en va pas de même pour les caractères spéciaux ou accentués utilisés par les autres langues (français, allemand, espagnol, italien...). Si certaines normes existent (par exemple ISO 8859), celles-ci ne sont pas systématiquement mises en oeuvre dans tous les systèmes informatiques.

Dans la mesure où **Tun SQL** permet à un système informatique (le PC) d'accéder à des données situées sur un autre système informatique (un SGBD sous UNIX), il a été nécessaire de fournir au sein de **Tun SQL** un mécanisme permettant de s'affranchir des différences de notation des caractères nationaux entre systèmes. Ce mécanisme permet à un PC d'afficher un caractère accentué (un 'é' par exemple) même s'il a été stocké sous un autre code dans le SGBD de la machine UNIX.

➤ Création des tables de conversion

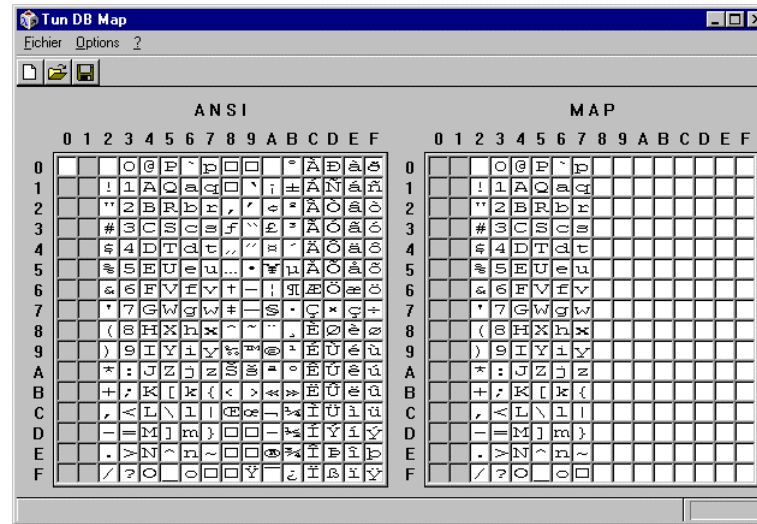
Le mécanisme offert par **Tun SQL** utilise des "tables de conversion" qui peuvent être créées ou mises à jour à l'aide de l'application **Tun DB Map**.



Lancez le programme en cliquant sur l'icône **Tun DB Map** du groupe **Data Access** (menu **Démarrer**, **Programmes**, **Esker Tun** sous Windows 95/98/2000 et Windows NT).



La fenêtre suivante apparaît :



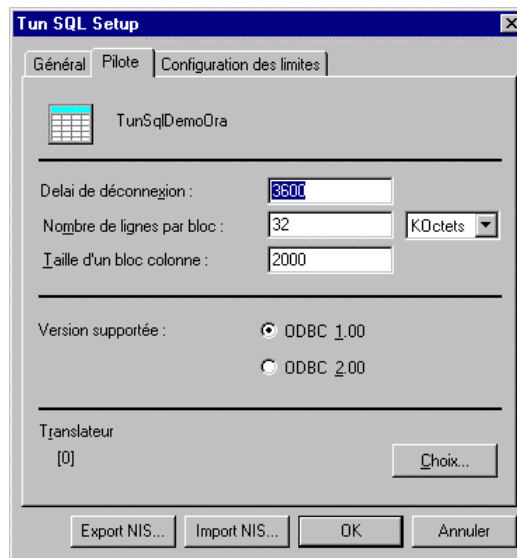
Le tableau de gauche contient tous les caractères disponibles sur le PC (ASCII et ASCII étendu). Le tableau de droite doit contenir les mêmes caractères mais avec les positions qui correspondent à la notation utilisée par le SGBD de la machine UNIX. Les 128 premières positions du tableau de droite sont déjà affectées car se sont les seules qui soient constantes d'un système à l'autre.

Pour affecter une des positions du tableau de droite il suffit de sélectionner un caractère dans le tableau de gauche et de le "tirer" vers une des positions du tableau de droite en maintenant le bouton de la souris enfoncé.

Il est possible de créer autant de tables de conversion que nécessaire en les sauvegardant dans des fichiers suffixés en ".ttt" à l'aide de l'option **Fichier** du menu général.

➤ Prise en compte des tables de conversion

Pour être prises en compte, les tables de conversion doivent être associées à une source de données à l'aide de l'écran fourni par **Tun SQL** en renseignant le groupe **Translateur** de la fenêtre de configuration du pilote ODBC :



Vous pouvez cliquer sur le bouton **Choix...** pour sélectionner le translateur ODBC.



C-ISAM

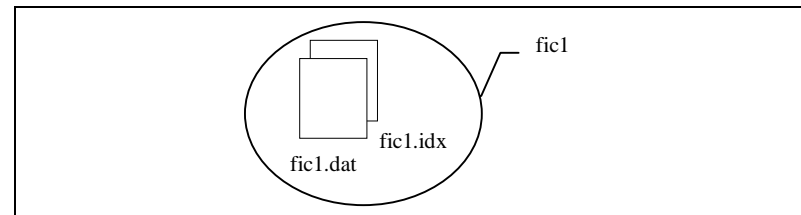
Présentation de C-ISAM

➤ Le système de fichiers C-ISAM

C-ISAM (Indexed Sequential Access Method) est une bibliothèque de fonctions C développées par Informix. Elle permet la gestion de fichiers séquentiels indexés (création de fichiers, insertion, suppression, lecture de données). C-ISAM inclut d'autres fonctionnalités telles que le verrouillage des fichiers et le support de transaction qui assurent l'intégrité des données. Ces fonctionnalités garantissent que l'information est accessible, cohérente et correctement exploitée.

C-ISAM utilise des types de données équivalents à ceux utilisés par le langage C. L'implémentation de ces types pour C-ISAM étant indépendant de la machine UNIX utilisée, la manière dont C-ISAM stocke les données peut être différente de celle dont ces données sont représentées durant l'exécution d'un programme. C-ISAM fournit des fonctions de conversion des formats d'exécution en format de stockage.

Chaque fichier C-ISAM est en fait l'association de deux fichiers : l'un contenant les données (fichier **.dat**), l'autre contenant l'index permettant de localiser l'information dans le fichier de données (fichier **.idx**). Ces deux fichiers sont toujours utilisés ensemble, comme un unique fichier logique C-ISAM.



➤ Les bases de données relationnelles et C-ISAM

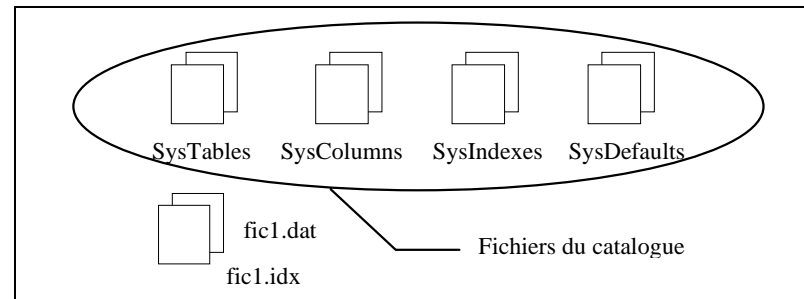
L'aspect séquentiel indexé des fichiers C-ISAM nécessitent pour les développeurs de connaître la structure des fichiers et d'utiliser les fichiers d'index pour accéder aux données.

Concevoir un système de bases de données autour de fichiers C-ISAM permet de s'affranchir de ces contraintes en intégrant la structure des fichiers indexés sous la forme de tables, de colonnes, d'index, au sein d'un catalogue.

➤ Principe de Tun SQL C-ISAM

Le principe de C-ISAM repose sur l'utilisation de routines C pour l'interrogation et la mise à jour des fichiers séquentiels. Le driver C-ISAM proposé par **Tun SQL** permet de voir les fichiers séquentiels comme une base de données relationnelle classique avec tables, champs, clés. En utilisant des ordres SQL classiques, l'utilisateur peut alors interroger ou mettre à jour sa base de données créée sur des fichiers C-ISAM. Le driver C-ISAM traduit ces ordres en fonctions C capables d'effectuer les opérations souhaitées sur les fichiers séquentiels.

Pour passer d'une vision des données séquentielle à une vision de type base de données relationnelle, le driver C-ISAM crée autour des fichiers de données et d'index C-ISAM classiques, des fichiers descriptifs de la base de données : le catalogue. Ces fichiers sont de type C-ISAM (fichier de données **.dat** et fichier d'index **.idx**) : SysTables, SysColumns, SysIndexes, SysDefaults.



L'outil **sqltools** de création et gestion sous UNIX des bases de données fondées sur C-ISAM, fonctionne selon le même principe : des ordres, de type SQL, permettent de construire la base de données et sont traduits en fonctions C afin d'être compris par le système de fichiers C-ISAM.

➤ Installation du driver C-ISAM

L'installation du driver **Tun SQL C-ISAM** est expliquée dans le "**Installation et configuration**". Reportez-vous à ce manuel pour procéder à cette opération.

Utilisation de sqltools

➤ Lancement de sqltools

Vous pouvez utiliser les fonctionnalités de **sqltools** en ligne ou au sein d'une interface semi-graphique (fenêtres, menus).

Connectez-vous au serveur UNIX sur lequel vous souhaitez travailler avec les fichiers C-ISAM. Nous vous recommandons de créer un login utilisateur pour l'accès aux fichiers C-ISAM.

Placez-vous dans le répertoire d'installation de **sqltools**, puis lancez l'appli en entrant la commande suivante :

```
sqltools  
pour un lancement en ligne,
```

```
sqltools -v  
pour l'utiliser sous interface graphique.
```

➤ Création de la base de données

La première étape consiste à créer la base de données qui contiendra les données issues des fichiers C-ISAM. Pour cela, procédez selon l'une des méthodes suivantes :

- Sélectionnez l'option **Database→Create** et entrez le nom de la base de données que vous voulez créer.

- Entrez la commande suivante dans la fenêtre inférieure (fenêtre Input):

```
create database "basename";
```

Cette commande crée dans le répertoire correspondant à la variable ISAM-PATH, un répertoire du nom de la base à l'extension ".ism". Voir "**Installation du driver C-ISAM**" dans "**Guide d'installation Tun**".

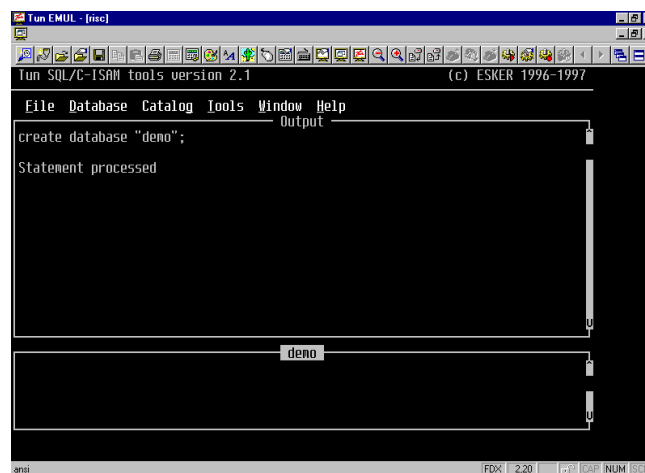
Exemple :

```
Create database "demo";
```

créé le répertoire demo.ism dans le répertoire /TunSql/bases si ISAM-PATH=/TunSql/bases

Ce répertoire contient les fichiers C-ISAM .dat et .idx descriptifs de la base : SysTables, SysColumns, SysIndexes et SysDefaults, soit au total 4 fichiers logiques C-ISAM et 8 fichiers physiques.

La fenêtre inférieure (fenêtre Input) prend le nom de la base de données :



Remarque

Vous pouvez exécuter des commandes shell directement sous **sqltools**. Entrez la commande précédée d'un point d'exclamation et suivie d'un point virgule dans la fenêtre inférieure (fenêtre Input).

Exemple :

```
!ls -a ;
```

Pour entrer des chaînes de caractères en majuscules, utilisez des guillemets.

Exemple :

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

➤ Connexion à une base de données existante

Si vous disposez déjà d'une base de données, vous pouvez effectuer des opérations sur cette table après vous y être connecté.

Pour cela, sélectionnez l'option **Database** ➔ **Connect** et entrez le nom de la base de données. La fenêtre inférieure (fenêtre Input) prend alors le nom de la base de données à laquelle vous êtes connecté.

➤ Création des tables

Lorsque la base de données est créée (répertoire ".ism" contenant les fichiers descriptifs de la table), vous pouvez alors créer les tables qui la constitue.

Cette étape peut être réalisée à partir d'un couple de fichiers C-ISAM existant (fichier de données **.dat** et fichier d'index **.idx**), ou sans existant. La commande utilisée alors est différente, ainsi que les précautions à prendre : commande **create table** pour créer le couple de fichiers C-ISAM en même temps que la table, commande **define table** pour créer une table à partir d'un couple de fichiers C-ISAM existants.

Création des tables sans existant

Pour créer le couple de fichiers C-ISAM en même temps que la table, entrez la commande suivante dans la fenêtre inférieure (fenêtre Input) portant le nom de la base de données :

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



Cette commande crée dans la base la table de nom "tablename" contenant les champs field1, field2,... de type type1, type2,... Voir la liste des types.

Les deux fichiers de données et d'index C-ISAM sont créés dans le répertoire de la base (**basename.ism**). Leur nom est composé des 7 premiers caractères du nom de la table, d'un numéro unique attribué automatiquement, et du suffixe ".dat" pour le fichier de données et ".idx" pour le fichier d'index. Si le nom de la table est inférieur à 7 caractères, il est complété par des "_".

Exemple :

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

*crée les fichiers **table1_100.dat** et **table1_100.idx**. La table contient un champ field1 de type longint, un champ field2 de type char, de longueur maximale 25, et un champ filler de type char de longueur maximale 30. La clé primaire de cette table est le champ field1.*

Création des tables à partir d'un couple de fichiers C-ISAM existant

Pour créer une table dans une base de données pour laquelle le couple de fichiers C-ISAM ".dat" et ".idx" existe déjà, entrez la commande suivante dans la fenêtre inférieure (fenêtre Input) portant le nom de la base de données :

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

Cette commande crée la table de nom "tablename" à partir des fichiers déjà existants **filename.dat** et **filename.idx**.

Remarque importante

Avant d'utiliser la table ainsi définie (par exemple, des ordres **select** sur cette table), vous devez avoir copié les fichiers spécifiés par l'option **file is** dans le répertoire de la base.

Cependant, l'ordre **define** peut être exécuté sans que ces fichiers ne soient encore copiés dans ce répertoire. Il est même recommandé si vous souhaitez créer un index pour cette table, de ne copier les fichiers qu'après avoir exécuté l'ordre **create index**.



➤ Création d'un index

Pour créer un index sur une à huit colonnes d'une table, entrez la commande suivante dans la fenêtre inférieure (fenêtre Input) portant le nom de la base de données :

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

Cette commande crée l'index "indexname" sur les colonnes field1 et field3 de la table de nom "tablename".

➤ Structure C

Chaque ordre passé à **sqltools** est traduit en langage C pour le système de fichiers C-ISAM. Pour visualiser la structure C d'une table, sélectionnez l'option **Catalog**→**GetCStruct**.

A titre d'exemple, l'ordre de création de la table **table1** correspond à la structure C suivante :

```
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 */
};
```

Dans cet exemple de structure C, apparaît un champ de réserve (unsigned char) d'une longueur de 1 octet. Ce champ est spécifique au système de gestion C-ISAM. Lors de la création d'une table par l'ordre **create table**, **sqltools** ajoute automatiquement ce champ de réserve à la table. Dans notre exemple, la structure C le prouve.

➤ Contrôle de la cohérence d'une table avec les fichiers C-ISAM existants

Lorsque vous créez une table à partir d'un couple de fichiers C-ISAM existant (ordre **define**), vous devez veiller à ce que la structure de la table que vous créez correspond bien à celle des fichiers C-ISAM.

A titre d'exemple, vous créez la table "table2" basée sur les fichiers C-ISAM **filename.dat** et **filename.idx**, créés en C. Ces fichiers correspondent à la structure d'enregistrement suivante :

- Un champ longint,
- Un champ char de longueur 25,
- Un champ char de longueur 30.

Si vous définissez la table "table2" de la manière suivante :

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

vous créez en fait une table dont les enregistrements ont pour structure :

- Un champ longint,
- Deux champs char de longueur 25,

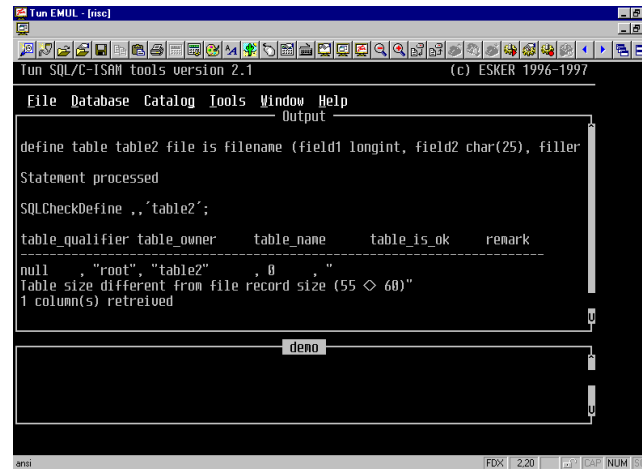
ce qui ne correspond pas à la structure proposée par les fichiers C-ISAM sur lesquels est basée la table.

Dans ce cas, il y a incohérence entre la table créée et les fichiers C-ISAM dont elle dépend.

Pour contrôler la cohérence d'une table avec les fichiers C-ISAM initiaux, sélectionnez l'option **Catalog→CheckDefine**. Pour contrôler la cohérence de l'ensemble des tables de la base, sélectionnez l'option **Tools→Check Catalog**.



Dans notre exemple, la fenêtre Output fournit les résultats suivants :



Un message apparaît :

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

Ce message traduit le fait que la table **table2** a été définie différemment des fichiers C-ISAM "filename" sur laquelle elle est basée. L'ordre **define** devrait être le suivant :

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

➤ Visualisation de la structure C d'une table

Vous pouvez visualiser la structure C d'une table, c'est-à-dire d'une part les différentes colonnes créées (nom, type et longueur des données), d'autre part les informations relatives à la gestion des données (clés primaires par exemple).

Pour cela, sélectionnez l'option **Catalog**→**GetCStruct** et entrez le nom de la table dont vous souhaitez visualiser la structure C. Pour visualiser la structure C de l'ensemble des tables de la base, sélectionnez l'option **Tools**→**List Structures**.

L'exemple ci-dessous montre le résultat de cette commande pour une table créée avec une clé primaire :

Ordre de création de la table :

```
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)
);
```

Structure C obtenue :

```
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) */
 lchar  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;
```

La première partie décrit la structure de la table (champs), la seconde indique la clé primaire et ses caractéristiques.

➤ Informations sur le catalogue

Vous pouvez obtenir différentes informations concernant le catalogue à l'aide du menu **Catalog**. Les options de ce menu informent notamment sur :

- **TypeInfo** : chaque type de données est identifié par un numéro, correspondant à la norme ODBC. Entrez le numéro du type de données voulu, ou tapez 0 pour obtenir la liste de tous les types.



- **Tables** : vous pouvez interroger le catalogue sur les tables qu'il contient, à partir du nom de l'utilisateur ayant créé la table, du nom de la table ou du type de table (table système, synonyme,...). Entrez le caractère % pour faire votre requête sur toutes les tables ou tous les types de tables.
- **Columns** : vous pouvez interroger le catalogue sur les colonnes qu'il contient, à partir du nom de l'utilisateur ayant créé la table, du nom de la table ou du nom de la colonne. Entrez le caractère % pour faire votre requête sur toutes les tables ou toutes les colonnes.
- **Statistics** : vous pouvez obtenir des statistiques sur les données du catalogue.
- **PrimaryKeys** : vous pouvez interroger le catalogue sur les clés primaires d'une table, à partir du nom de l'utilisateur ayant créé la table ou du nom de la table. Entrez le caractère % pour faire votre requête sur toutes les tables.

Pour obtenir de plus amples informations concernant les options du menu **Catalog**, consultez le manuel de référence au système de catalogue de la norme ODBC.

➤ **Modification d'une table**

La longueur des enregistrements d'une table est fixée lors de sa création. Par conséquent, vous ne pouvez ajouter ou supprimer des enregistrements si la longueur totale en est affectée.

Si vous souhaitez modifier la structure d'une table, vous devez d'abord procéder à la suppression de la table en respectant bien les précautions détaillées au paragraphe "**Suppression d'une table**".

➤ **Suppression d'une table**

Vous pouvez supprimer une table du catalogue selon deux méthodes :

- Vous avez créé la table à l'aide de l'ordre **create table** : utilisez l'ordre **drop table** pour la supprimer. Attention : cet ordre a pour conséquence d'enlever toute référence à cette table au sein des fichiers du catalogue, et de supprimer le couple de fichiers C-ISAM associé à la table.

- Vous avez défini la table à l'aide de l'ordre **define table** : utilisez l'ordre **undefine table** pour annuler l'ordre **define**. Cet ordre a pour conséquence uniquement d'enlever toute référence à cette table au sein des fichiers du catalogue.

Remarque

Il est possible d'utiliser l'ordre **drop table** pour une table définie par **define table**. Cependant, sachez que cet ordre de suppression supprimera le couple de fichiers C-ISAM spécifiés par l'ordre **file is**, s'ils sont déjà présents dans le répertoire de la base de données. Il est donc recommandé d'être prudent avec l'ordre **drop table** si vous partez de fichiers C-ISAM existants.

Dans le cas où vous souhaitez conserver les fichiers C-ISAM associés à la table supprimée par l'ordre **drop table**, vous devez impérativement en avoir fait une copie préalable dans un répertoire différent. Ainsi, vous pourrez après suppression de la table, récupérer cette sauvegarde pour d'autres usages.

➤ **Maintenance des fichiers C-ISAM**

Lorsque vous créez des tables sur des fichiers C-ISAM existants (ordre **define table**), vous devez copier ces fichiers dans le répertoire de la base de données pour pouvoir utiliser les tables.

Cependant, si ces fichiers sont utilisés et mis à jour par ailleurs (autres applications), il est intéressant de pouvoir disposer de ces mises à jour au sein de votre base de données C-ISAM. Pour cela, vous devez établir un lien symbolique des fichiers C-ISAM existants vers le répertoire de la base de données, plutôt que d'en faire une copie.

Exemple :

*Vous avez créé une base de données **dbtest** dans le répertoire **/TunSql**. Vous faites appel pour les tables de cette base de données aux fichiers **filename.dat** et **filename.idx**, qui se trouvent dans le répertoire **/data**, utilisés par d'autres applications.*

*Vous établissez donc un lien symbolique de ces fichiers vers le répertoire **/TunSql/dbtest.ism** (répertoire de la base de données) en exécutant la commande suivante :*

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



➤ **Suppression de la base de données**

Pour supprimer une base de données, sélectionnez l'option **Database→Drop** et entrez le nom de la base à supprimer, ou entrez la commande suivante dans la fenêtre inférieure (fenêtre Input) :

```
drop database basename
```

➤ **Sauvegarde des résultats**

Vous pouvez sauvegarder les résultats obtenus dans la fenêtre supérieure (fenêtre Output), au sein d'un fichier texte d'extension ".res".

Pour cela, sélectionnez l'option **File→Save as...** et choisissez la destination et le nom du fichier de sauvegarde.

➤ **Lancement d'un script**

A l'aide de l'option **File→Execute**, vous pouvez exécuter un script SQL, à condition de respecter les ordres SQL acceptés par **sqltools**.

PARTIE 2
REVAMPING DE
BASES DE DONNEES

LE REVAMPING

Les bases de données virtuelles

Les systèmes de gestion de bases de données relationnelles (SGBDR) représentent aujourd'hui la quasi-totalité des structures de données. Les bases de données permettent de stocker les données de l'entreprise qui y sont mises à jour par l'intermédiaire d'applications spécifiques. La masse de données ainsi rassemblée intéresse parallèlement de nombreux utilisateurs qui souhaitent puiser dans ces "réservoirs" les informations nécessaires à leur métier (tableaux de bord, statistiques, aide à la décision). Le langage SQL est utilisé pour mettre à jour et interroger les bases de données.

Cependant, la structure des bases de données au sein du système d'informations rend difficile l'accès à l'information à tous les niveaux de l'entreprise :

- Le nombre de tables ou de champs enregistrés dans les bases est considérable et bien trop important pour chaque utilisateur final. Seules quelques unes des données l'intéressent réellement.
- La structure des bases est toujours complexe, et nécessite beaucoup d'expérience pour s'y retrouver rapidement.
- L'environnement informatique des bases de données est peu convivial. Par exemple, les noms des tables ou des champs qui composent les bases sont rarement exprimés en langage clair.
- Le traitement des données nécessite de connaître le langage SQL pour interroger les bases et obtenir les résultats souhaités.

Plusieurs étapes d'évolution ont déjà été franchies pour réduire ces obstacles et démocratiser l'accès aux bases de données (par exemple l'apparition d'une interface graphique dans les outils d'interrogation des bases de données).

L'étape suivante vise à affranchir quasi-totalement l'utilisateur final de la connaissance technique des bases de données, en mettant à sa disposition uniquement l'information dont il a besoin sous la forme la plus adaptée à son environnement.

Les conséquences d'une telle évolution sont :

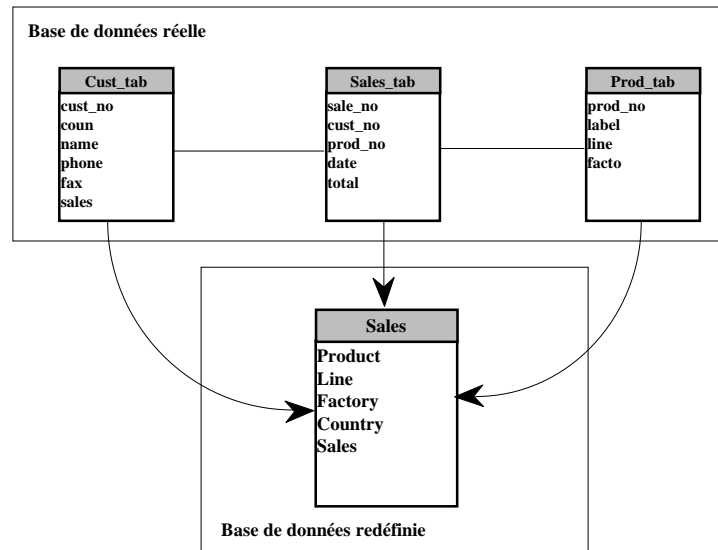
- Une productivité améliorée : l'utilisateur final devient autonome dans le traitement des données, l'analyse et la prise de décision sont plus rapides car facilitées.
- Une pertinence accrue de l'information : en ne disposant que des données dont il a besoin et qu'il maîtrise, l'utilisateur aiguise sa capacité d'analyse et de synthèse, et affine ses résultats.

Le principe consiste à construire, à partir de la base de données existante, une structure virtuelle de base de données adaptée au contexte de l'utilisateur final. Bien que n'existant pas en tant que base de données réelle, cette nouvelle structure est vue par l'utilisateur comme une base de données traditionnelle, dont les tables et les champs sont cependant conformes à ses besoins : cette base ne contient que l'information dont l'utilisateur a réellement besoin pour ses analyses, et sous la forme qui convient à son exploitation (noms des données intelligibles, formules de calcul pré-définies).

La base de données ainsi redéfinie est mise au point par un administrateur qui est chargé d'en configurer les tables et les champs à partir des bases de données réelles.



Exemple :



Dans l'exemple présenté, la base de données réelle contient trois tables : "Cust_tab" (table des clients), "Sales_tab" (table des ventes) et "Prod_tab" (table des produits).

L'administrateur définit une table virtuelle permettant d'obtenir le résultat des ventes selon les produits, les lignes de produits, les sites de fabrication, les pays dans lesquels ont été menées ces ventes.

Cette table virtuelle nommée "Sales" contient les champs suivants :

- Product (champ réel : "prod_tab.label")
- Line (champ réel : "prod_tab.line")
- Factory (champ réel : "prod_tab.facto")
- Country (champ réel : "cust_tab.coun")
- Sales (champ réel : "sales_tab.total")

Elle effectue une jointure entre les tables "Prod_tab" et "Sales_tab" par le champ "prod_no", et une jointure entre les tables "Sales_tab" et "Cust_tab" par le champ "cust_no".

Intégration du revamping dans Tun SQL

Tun SQL intègre l'administration et l'exploitation de bases de données virtuelles grâce aux deux composants suivants :

- Le module **Tun DB Revamp** d'administration des bases de données, destiné à leur redéfinition.
- Le driver ODBC virtuel grâce auquel l'utilisateur accède aux bases de données redéfinies par l'administrateur.

➤ Module d'administration Tun DB Revamp

Le principe de la base de données virtuelle dans **Tun SQL** est de mettre à disposition de l'utilisateur final un ensemble d'informations configurées selon son contexte, regroupées au sein d'une entité particulière appelée "environnement".

Grâce à une interface graphique intuitive, l'administrateur définit autant d'environnements qu'il le souhaite, en fonction de la variété des utilisateurs ou groupes d'utilisateurs. L'environnement est orienté "métier" : un comptable ne verra que les tables ayant trait à la comptabilité, un commercial ne verra que les tables ayant trait à son activité.

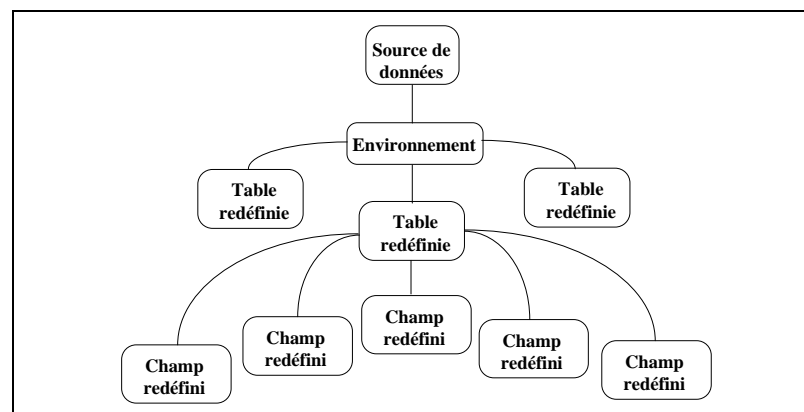
Chaque environnement pourra être vu comme une source de données particulière vis-à-vis du frontal ODBC qui effectue la requête (voir schéma de l'architecture dans le paragraphe "**Driver ODBC virtuel**").

Le modèle de données d'une base virtuelle est le suivant :

- Un ou plusieurs environnements définis à partir d'une source de données réelle, contenant une sélection de certaines tables de la base de données réelle selon les besoins des utilisateurs finals.
- Les tables définies dans un environnement sont soit des tables "natives" de la base de données réelle, soit le résultat de jointures entre deux ou plusieurs tables (notion de vue).
- Chaque table contient les champs nécessaires à l'utilisateur final, et uniquement ceux-là.
- Les champs redéfinis sont soit des champs existant déjà dans les tables réelles, soit des champs calculés qui simplifieront encore l'utilisation finale de la base de données.



- Les tables et champs redéfinis peuvent être rebaptisés avec des noms plus explicites pour l'utilisateur final (exemples : 'Cust_tab' devient 'Table des clients' ; "Cust_no" devient "Numéro de client").



Les tables redéfinies dans un environnement n'ont pas d'existence physique dans la base de données. Cependant, la redéfinition d'une base de données est stockée dans trois tables supplémentaires créées dans cette base et représentant :

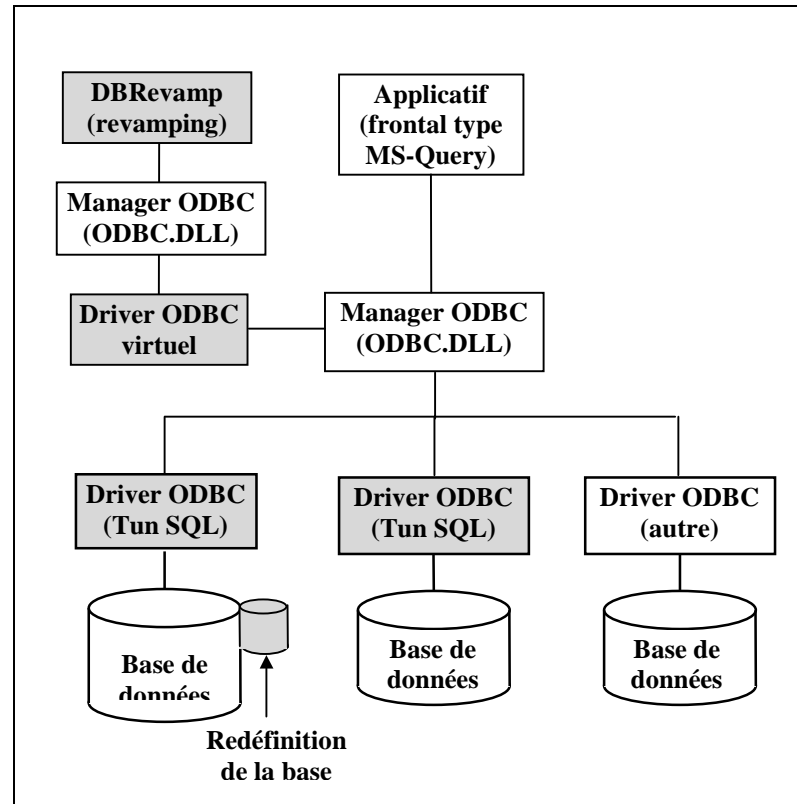
- La table des environnements, chaque environnement étant défini par son nom et son descriptif.
- La table des tables redéfinies, chaque table étant définie par son nom, son descriptif et son environnement d'appartenance.
- La table des champs redéfinis, chaque champ étant défini par son nom, son descriptif, son origine (champ existant, donnée calculée, concaténation de données) et sa table virtuelle d'appartenance.

Après redéfinition, une base de données contiendra toujours ces trois tables supplémentaires.

➤ Driver ODBC virtuel

Pour l'utilisateur final, l'interrogation d'une base de données virtuelle est semblable à celle d'une base de données réelle, en lecture uniquement. Cette transparence est due à l'intégration dans **Tun SQL** d'un driver ODBC spécifique aux bases de données virtuelles.

Lorsque le manager ODBC (**ODBC.DLL**) reçoit des requêtes exprimées dans un environnement, il les transmet au driver ODBC virtuel qui se charge alors de les traduire en requêtes exprimées selon le schéma de la base de données réelle. Le driver ODBC virtuel retransmet ensuite les requêtes ainsi traduites au manager ODBC qui les redirige alors vers le driver ODBC classique correspondant à la base de données réelle interrogée.



UTILISATION GENERALE DE TUN DB REVAMP

Remarques générales

➤ Choisir la langue de travail

Pour choisir la langue dans laquelle vous souhaitez utiliser **Tun DB Revamp**, cliquez sur l'option ?→**Langue** et sélectionnez la langue de votre choix.

➤ Modifier l'affichage



Pour modifier l'affichage des barres de la fenêtre de **Tun DB Revamp** :

- Sélectionnez ou désélectionnez l'option **Affichage→Barre d'outils** du menu général pour afficher ou effacer la barre de boutons :
- Sélectionnez ou désélectionnez l'option **Affichage→Barre d'état** du menu général pour afficher ou effacer la barre d'état.
- Sélectionnez ou désélectionnez l'option **Affichage→Boîte de propriétés** du menu général pour afficher ou effacer la boîte à onglets des propriétés des objets.

➤ Copier un objet


Pour copier un objet d'un endroit à l'autre, procédez selon l'une des méthodes suivantes :

1. Utilisez la fonction **drag and drop** en sélectionnant l'objet à copier, et en le déposant sur l'objet dans lequel vous souhaitez le placer.
2. Utilisez l'option **Edition→Copier** du menu général pour copier l'objet sélectionné, puis l'option **Edition→Coller** du menu général pour le déposer à l'endroit voulu.

3. Utilisez l'option **Copier** du menu contextuel de l'objet à copier, puis l'option **Coller** du menu contextuel de l'objet dans lequel vous souhaitez le placer.
4. Utilisez les raccourcis clavier **Ctrl+C** pour copier, **Ctrl+V** pour coller.
5. Utilisez les boutons  pour copier et  pour coller.

➤ Supprimer un objet

Pour supprimer un objet, sélectionnez-le puis procédez selon l'une des méthodes suivantes :

1. Utilisez l'option **Edition→Effacer** du menu général.
2. Utilisez l'option **Effacer** du menu contextuel de l'objet sélectionné.
3. Utilisez la touche **Suppr** ou **Del** de votre clavier.
4. Cliquez dans la barre de boutons sur le bouton .

➤ Renommer un objet


Pour renommer un objet, sélectionnez-le puis procédez selon l'une des méthodes suivantes :

1. Utilisez la boîte à onglets **Général** relative à l'objet.
2. Utilisez la touche **F2** de votre clavier et remplacez l'ancien nom par le nouveau.
3. Cliquez une deuxième fois sur l'objet et procédez comme au deuxième point.

➤ Enregistrer des modifications

Pour enregistrer des modifications dans les valeurs d'une propriété, utilisez la touche **Entrée** de votre clavier en vous assurant que le curseur est bien situé dans la boîte de dialogue concernée, ou bien utilisez le bouton **Appliquer** de la boîte de dialogue.

➤ Obtenir de l'aide


Pour accéder à l'aide en ligne ou pour en savoir plus sur **Tun DB Revamp**, cliquez sur l'option **?→A propos de DBRevamp** du menu général, ou cliquez dans la barre de boutons sur le bouton .



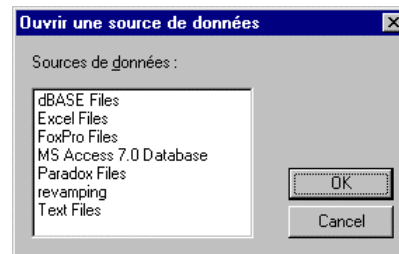
➤ Quitter Tun DB Revamp

Pour quitter l'application, cliquez sur l'option **Fichier→Quitter** du menu général.

Importation des environnements d'une source de données

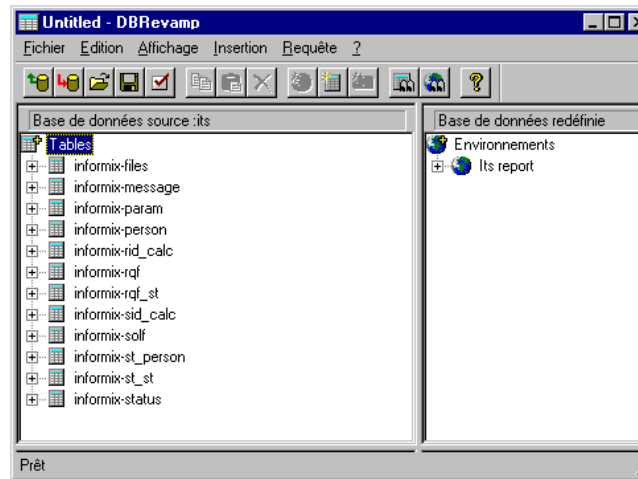
Pour pouvoir procéder à la redéfinition d'une base de données réelle, vous devez sélectionner la source de données correspondante à cette base. Pour cela, sélectionnez l'option **Fichier→Importer...** du menu général, ou bien cliquez dans la barre de boutons sur le bouton .

La boîte de dialogue suivante apparaît :



Elle contient la liste des sources de données déclarées sur le PC. Pour créer une source de données, voir "**Création d'une source de données**". Les sources de données virtuelles, obtenues par revamping d'une base de données réelle, ne pouvant pas être à leur tour redéfinies, elles n'apparaissent pas dans cette liste. Cependant, elles apparaissent dans la liste des sources de données mises à disposition de l'utilisateur final dans toute application Windows les exploitant (exemple : **MS Query**).

Sélectionnez la source de données de votre choix. Une fenêtre **Tun DB Revamp** de ce type apparaît :




Dans la partie gauche apparaissent les tables de la base de données réelle.

Si la base de données choisie n'a jamais été redéfinie par **Tun DB Revamp**, la fenêtre de droite affiche un seul environnement vide nommé **Nouvel environnement**, qui constitue le premier environnement que vous pouvez configurer.

En revanche, si la base de données a déjà été redéfinie par **Tun DB Revamp** (il s'agit dans ce cas d'une mise à jour de la base de données virtuelle), la partie droite de la fenêtre fait apparaître les environnements déjà créés et leur contenu.

Création d'un environnement



Pour définir un nouvel environnement relatif à la source de données sélectionnée, sélectionnez la racine des environnements (nommée **Environnements**) puis cliquez sur **Insertion**→**Nouvel environnement** du menu principal ou bien cliquez dans la barre de boutons sur le bouton .




Affectez le nom et éventuellement la description de votre choix à cet environnement.

Création d'une table virtuelle



Pour créer une table virtuelle dans un environnement, sélectionnez l'environnement puis procédez comme suit :

- Cliquez sur **Insertion**→**Nouvelle table** du menu principal ou bien choisissez l'option **Nouvelle table** du menu contextuel de l'environnement ou bien encore cliquez dans la barre de boutons sur le bouton .
- Affichez si tel n'est pas le cas, la boîte de propriétés de la table nouvellement créée en sélectionnant l'option **Affichage**→**Boîte de propriétés**.
- Dans la boîte à onglets **Général**, affectez à cette table le nom et éventuellement la description de votre choix. Pour renommer la table, vous pouvez également utiliser la touche de fonction **F2** après avoir sélectionné la table.

Si vous souhaitez que la table virtuelle contienne tout ou partie d'une table réelle, vous pouvez copier cette table réelle dans l'environnement de votre choix : tous les champs contenus dans cette table seront eux aussi copiés. Pour cela :

- Procédez selon l'une des méthodes proposées en introduction (**drag and drop**, **Copier/Coller**, raccourci clavier ou bouton), en sélectionnant la table réelle dans la base de données source, et en la copiant dans l'environnement destinataire.
- Supprimez les champs dont vous ne souhaitez pas la présence dans la table virtuelle, ou modifiez-les comme expliqué au paragraphe "**Création d'un champ**".
- Si vous le souhaitez, modifiez les noms des entités copiées (tables et champs) et affectez-leur une description dans la boîte à onglets **Général** qui leur correspond.

Création d'un champ



Dans une table virtuelle, vous pouvez :

- Placer un champ qui existe déjà dans une table de la base de données réelle, sans modifier sa définition.
- Créer un nouveau champ virtuel, à partir des champs de la base de données réelle.


➤ Champ existant

Si vous souhaitez placer dans la table virtuelle un champ existant dans une table de la base de données réelle, vous pouvez copier ce champ dans la table virtuelle. Pour cela :

- Procédez selon l'une des méthodes proposées en introduction (**drag and drop**, **Copier/Coller**, raccourci clavier ou bouton), en sélectionnant le champ dans la table de la base de données réelle, et en le déposant dans la table virtuelle de la base de données redéfinie.
- Si vous le souhaitez, modifiez le nom de ce champ et affectez-lui une description dans la boîte à onglets **Général** qui lui correspond (ou bien utilisez la touche de fonction **F2** pour le renommer).

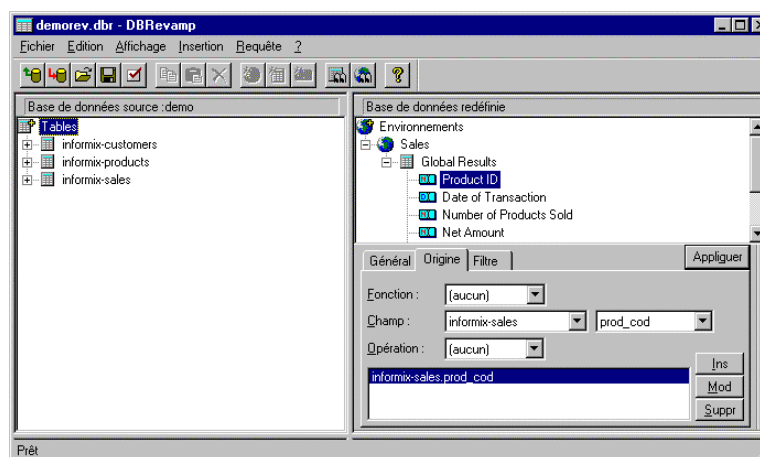
➤ Nouveau champ

Pour définir un nouveau champ dans une table virtuelle, sélectionnez la table virtuelle puis procédez comme suit :

- Cliquez sur **Insertion**→**Nouveau champ** du menu général ou bien choisissez l'option **Nouveau champ** du menu contextuel de la table ou bien encore cliquez sur le bouton  dans la barre de boutons.
- Affichez si tel n'est pas le cas, la boîte de propriétés de la table nouvellement créée en sélectionnant l'option **Affichage**→**Boîte de propriétés**. Dans la boîte à onglet **Général**, affectez le nom et éventuellement la description de votre choix à ce champ.
- Cliquez sur l'onglet **Origine**. Vous pouvez ensuite :
 - Affecter une fonction à ce champ : sélectionnez la fonction de votre choix dans la liste déroulante de l'option **Fonction**. Les fonctions possibles sont : **Somme**, **Min**, **Max**, **Nombre**, **Moyenne**, ou aucune.



- Affecter à ce champ ou à la fonction sélectionnée au-dessus la valeur d'un champ existant dans une table de la base de données réelle : sélectionnez la table et le champ réels de votre choix dans chacune des listes déroulantes de l'option **Champ**.
- Affecter une opération au champ sélectionné au-dessus : sélectionnez l'opérateur de votre choix dans la liste déroulante de l'option **Opération**. Les opérations possibles sont : +, -, *, /, ou aucune. La concaténation de caractères est possible par l'opérateur +.
- Cliquez ensuite sur le bouton **Ins** pour ajouter cette sélection à la définition du champ.



Exemple 1 :

Vous disposez d'une table réelle "res_tab" contenant quatre champs res1, res2, res3 et res4, correspondant au résultat des ventes sur chacun des quatre trimestres d'une année. Vous souhaitez définir le champ "Résultat" dans une table de votre base de données virtuelle, permettant d'obtenir la somme des quatre champs réels.

Dans la boîte à onglets **Origine** du champ **Résultat** :

- Sélectionnez la table "res_tab" et le champ "res1" dans chacune des listes déroulantes de l'option **Champ**.
- Sélectionnez l'opérateur + dans la liste déroulante de l'option **Opération**.

- Cliquez sur le bouton **Mod** pour remplacer l'entrée sélectionnée par défaut. La nouvelle entrée est "res_tab.res1 +".
- Sélectionnez ensuite la table "res_tab" et le champ "res2" dans chacune des listes déroulantes de l'option **Champ**.
- Sélectionnez à nouveau l'opérateur + dans la liste déroulante de l'option **Opération**.
- Cliquez sur le bouton **Ins** pour ajouter l'entrée ainsi constituée. La nouvelle entrée est "res_tab.res2 +".
- Procédez de même pour le champ "res3".
- Pour le champ "res4", sélectionnez l'opérateur "aucun" à la place de "+".
- Le champ **Résultat** est défini à la fin par la liste suivante :

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

ce qui signifie que le champ **Résultat** a pour valeur la somme des quatre champs "res1", "res2", "res3" et "res4".

Exemple 2 :

Vous souhaitez que le champ **Résultat** donne comme valeur la somme de toutes les ventes de l'année. Vous devez pour cela, sommer d'abord tous les champs **res1**, tous les champs **res2**, ... puis summez ces 4 résultats

Dans la boîte à onglets **Origine** du champ **Résultat** :

- Sélectionnez la fonction **Somme** dans la liste déroulante du champ **Fonction**.
- Sélectionnez la table "res_tab" et le champ "res1" dans chacune des listes déroulantes de l'option **Champ**.
- Sélectionnez l'opérateur + dans la liste déroulante de l'option **Opération**.
- Cliquez sur le bouton **Mod** pour remplacer l'entrée sélectionnée par défaut. La nouvelle entrée est "res_tab.res1 +".
- Faites de même pour les champs "res2" et "res3", et sélectionnez pour le champ "res4" l'opérateur "aucun" au lieu de l'opérateur "+" de l'option **Opération**.



Vous pouvez modifier un élément de la définition d'un champ en utilisant le bouton **Mod** (l'élément en surbrillance est remplacé par les valeurs sélectionnées au-dessus). Cliquez sur le bouton **Suppr** pour supprimer l'élément en surbrillance.

Lorsque la définition du champ est complète, cliquez sur le bouton **Appliquer** pour la prendre en compte.

Dès qu'un nouveau champ virtuel est créé, veillez à définir les éventuelles jointures entre la ou les tables réelles utilisées pour constituer la table virtuelle. Voir paragraphe "**Liens entre tables réelles**".

Pour vérifier immédiatement que le calcul que vous attribuez au champ créé correspond à ce que vous recherchez, utilisez la fonction d'interrogation de **Tun DB Revamp**. Voir "**Interrogation des bases de données réelle et virtuelle**".

Affectation de filtres à un champ

La définition d'un champ virtuel peut être complétée par un filtre, c'est-à-dire par l'attribution d'un critère selon lequel la valeur du champ sera calculée. Ce filtre correspond aux critères éventuels soumis dans une requête (par exemple dans **MS Query**) afin de limiter la requête à ces critères.

Exemple :

Vous souhaitez obtenir la somme des champs "res1" lorsque le champ "res2" est supérieur à une certaine valeur. La restriction sur le champ "res2" est un filtre.

Tun DB Revamp permet d'attribuer à tout champ virtuel un filtre qui sera pris en compte tel quel lors de l'exploitation par l'utilisateur final de ce champ. Un filtre peut être :

- Statique : la valeur filtrante est fixée.
- Dynamique : c'est l'utilisateur qui entrera la valeur de son choix au moment de sa requête.

Pour attribuer un filtre à un champ virtuel, sélectionnez ce champ puis depuis la boîte de propriétés, sélectionnez l'onglet **Filtre**. Procédez ensuite de la manière suivante :

- Entrez un label au filtre que vous définissez dans le champ **Etiquette**. Dans le cas d'un filtre statique, ce label est facultatif. En revanche, dans le cas d'un filtre dynamique, ce label doit indiquer à l'utilisateur sur quoi porte le filtre pour lequel une valeur lui est demandée.
- Sélectionnez ensuite la table et le champ sur lesquels vous voulez appliquer ce filtre dans chacune des listes déroulantes de l'option **Champ**.
- Sélectionnez l'opérateur de comparaison dans la liste déroulante du champ **Comp**.
- Dans le cas d'un filtre statique, entrez la valeur du filtre dans le champ **Valeur**. Dans le cas d'un filtre dynamique, entrez la valeur ?.
- Cliquez sur le bouton **Ins** pour insérer le critère ainsi défini.

Vous pouvez créer une combinaison de critères en définissant plusieurs critères différents de la même manière que précédemment. Pour chacun de ces critères supplémentaires, sélectionnez l'opérateur logique **Et** ou **Ou** pour l'ajouter aux autres critères.

Pour vérifier immédiatement que le filtre que vous attribuez au champ créé correspond à ce que vous recherchez, utilisez la fonction d'interrogation de **Tun DB Revamp**. Voir "**Interrogation des bases de données réelle et virtuelle**".

Si le filtre est dynamique, toute requête sur le champ virtuel fera apparaître une fenêtre de ce type :

The screenshot shows a dialog box with the title "Veuillez entrer les critères suivants". It contains two rows of criteria. The first row is for "Code du pays" with an equals sign operator and a "Valeurs..." button. The second row is for "Taux de natalité mini" with a greater-than operator and a "Valeurs..." button. At the bottom, there are "Annuler" and "OK" buttons.



L'utilisateur doit y entrer la valeur demandée afin que le filtre puisse s'appliquer au champ virtuel. Cliquez sur le bouton **Valeurs...** pour afficher la liste des valeurs possibles pour le champ considéré.

Liens entre tables réelles

Les champs définis dans une table virtuelle sont obtenus indifféremment à partir d'une table ou une autre de la base de données réelle.

Pour chaque table virtuelle, il est indispensable de définir les liens entre les tables réelles dont sont extraits les champs qui la constitue. La définition de ces liens permettra d'effectuer les jointures entre tables réelles, lors de l'interrogation de la base de données virtuelle par l'utilisateur final. Ces liens peuvent être directs ou indirects (lien d'une table à l'autre, ou lien de tables en tables).

➤ Définition des liens

Le plus simple est de procéder au fur et à mesure de la définition des champs dans la table virtuelle. Toute table réelle sollicitée pour la définition d'un champ, doit être reliée directement ou indirectement aux autres tables réelles utilisées par la table virtuelle.

Pour définir les liens entre tables réelles pour une même table virtuelle, sélectionnez celle-ci puis procédez comme suit :

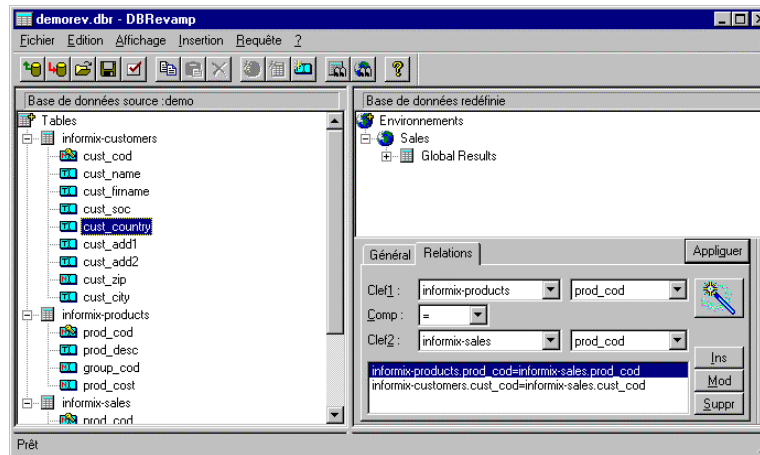
- Cliquez sur l'onglet **Relations** de la table virtuelle concernée.
- Sélectionnez pour chacune des tables réelles, son nom et le champ servant à la jointure avec l'autre table, en utilisant les listes déroulantes des champs **Clef1** pour la première table réelle, et **Clef2** pour la seconde table réelle.

Attention !

Les noms des deux champs reliant les tables peuvent être différents, même s'ils représentent la même donnée.

- Sélectionnez l'opérateur de comparaison de votre choix dans la liste déroulante de l'option **Comp**.

- Cliquez sur le bouton **Ins** pour ajouter ce lien à la liste des liens de la table virtuelle.



Vous pouvez modifier un lien en utilisant le bouton **Mod** après avoir modifié les valeurs de l'élément en surbrillance. Cliquez sur le bouton **Suppr** pour supprimer l'élément en surbrillance.


Cliquez sur le bouton **Appliquer** pour prendre en compte la liste des liens ainsi définie.

➤ Vérification des liens

Tun DB Revamp dispose d'une fonction de vérification de la cohérence des liens définis par l'administrateur entre toutes les tables réelles sollicitées pour construire une table virtuelle.

Pour chaque table virtuelle, vous pouvez vérifier très simplement si toutes les tables réelles concernées sont reliées entre elles, et si les liens définis forment un tout cohérent.

Pour cela, cliquez dans la boîte à onglets **Relations** de la table virtuelle

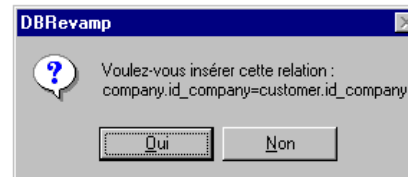
sur le bouton "magique"  .

Tun DB Revamp examine alors l'ensemble des liens définis par l'administrateur, et détecte l'absence éventuelle de liens directs ou indirects, qui isole certaines tables des autres.



Lorsqu'un lien manque entre deux tables, **Tun DB Revamp** tente de relier ces tables par deux champs de même nom.

Si ces deux champs existent, **Tun DB Revamp** propose à l'administrateur de définir le lien de cette manière.



Dans la plupart des cas, ce lien sera le bon. Cependant, si vous considérez que les deux champs proposés par **Tun DB Revamp** ne doivent pas constituer le lien entre les tables, définissez vous-même ce lien en procédant comme au paragraphe **Définition des liens**.

En revanche, si deux tables non reliées ne possèdent pas de champ avec le même nom, **Tun DB Revamp** fournit la liste des tables non reliées.





Dans ce cas, définissez vous-même ce lien en procédant comme au paragraphe "**Définition des liens**".

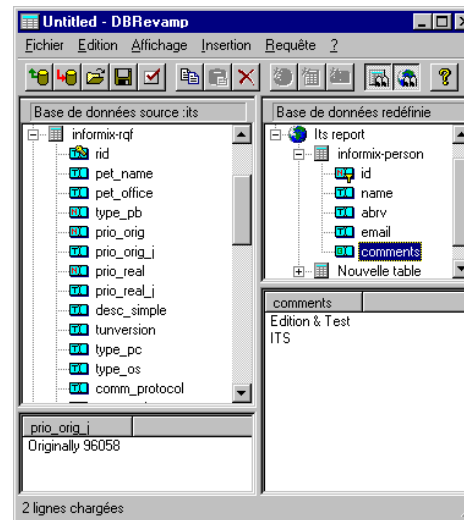
Interrogation des bases de données réelle et virtuelle

Tun DB Revamp intègre une fonction de requête disponible sur les tables et champs de la base de données réelle, et sur les tables et champs de la base de données virtuelle.

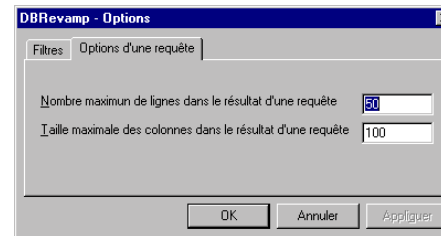
Cette fonction permet de connaître immédiatement depuis **Tun DB Revamp**, sans nécessaire passer par un outil de requête type **MS Query**, le contenu d'une table ou d'un champ dans la base de données réelle ou dans la base de données virtuelle.

Pour interroger une table (respectivement un champ) de la base de données réelle, sélectionnez l'option **Requête→Source** (respectivement **Requête→Environnement**) du menu général ou cliquez sur le bouton  (respectivement ) de la barre d'outils.

Une fenêtre s'affiche en-dessous de la fenêtre principale de la source de données réelle ou de la source de données virtuelle selon l'option choisie, les deux cas pouvant être simultanés.



Pour limiter le nombre d'enregistrements qui s'affichent lors de l'interrogation d'une table ou d'un champ, ainsi que la largeur des colonnes, sélectionnez l'option **Affichage**→**Options...** du menu général, puis l'onglet **Options d'une requête**.



Entrez le nombre maximum d'enregistrements à afficher dans le champ **Nombre maximum de lignes dans le résultat d'une requête**.

Entrez la largeur maximale des colonnes à afficher dans le champ **Taille maximale des colonnes dans le résultat d'une requête**.

Remarque

Si des limites ont été définies lors de la configuration de la source de données (voir "**Création d'une source de données**"), celles-ci sont prises en compte prioritairement par rapport aux options d'interrogation choisies dans **Tun DB Revamp**.

Exemple :

Une limite variable entre 11 et 15 lignes a été fixée lors de la configuration de la source de données réelle utilisée.


L'interrogation d'une table de la base de données réelle contenant plus de 11 enregistrements, provoque l'affichage d'une boîte de dialogue d'avertissement de ce type :



Le message d'avertissement varie selon le type de limites imposé.

Validation d'un environnement

Tun DB Revamp dispose d'une fonction permettant de vérifier la cohérence entre le contenu des environnements créés par l'administrateur et le contenu de la base de données réelle utilisée. Ceci est utile notamment lorsque des modifications ont été effectuées dans la structure de la base de données réelle (modification ou suppression d'un champ par exemple), sans que l'administrateur ne les ait prises en compte dans la base de données virtuelle.

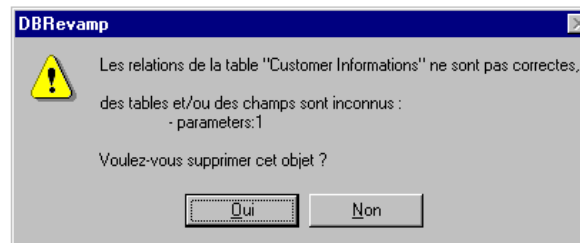
Pour utiliser cette fonction, il faut valider l'environnement avant de l'exporter, afin d'éviter toute incohérence possible. Pour cela, utilisez l'option **Fichier**→**Valider environnement...** du menu général, ou bien cliquez dans la barre de boutons sur le bouton .

A titre d'exemple, prenons le cas suivant où la table "parameters:1" et tous ses champs ont été copiés dans l'environnement "Marketing". Chaque champ de la table virtuelle ainsi créée a pour origine la table "parameters:1".

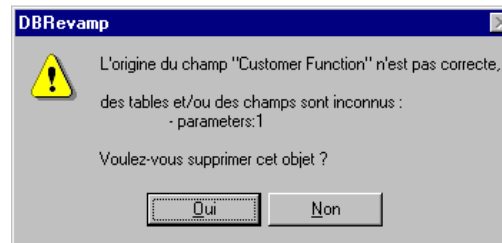
Si par la suite, cette table logique est supprimée de la base de données réelle, les champs de la table virtuelle n'auront plus de table origine. De même, toute table virtuelle ayant dans ces relations, une référence à la table "parameters:1", deviendra incohérente.

A la demande de validation de l'environnement, **Tun DB Revamp** :

- *Signale les tables virtuelles dont une ou plusieurs relations fait référence à la table supprimée, et propose de les supprimer. Dans ce cas, préférez ne pas les détruire et supprimez plutôt les champs dont l'origine est dans la table supprimée.*



- Signale les champs virtuels dont l'origine est dans la table supprimée, et propose de les supprimer.




Lorsqu'aucune incohérence n'a été détectée, la fenêtre suivante apparaît :



Exportation des environnements d'une source de données

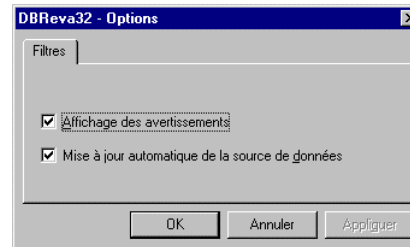
La base de données redéfinie par ses environnements doit être exportée du PC vers le serveur pour être mise à disposition des utilisateurs.

Pour exporter les environnements de la source de données du PC administrateur vers le serveur, utilisez l'option **Fichier**→**Exporter...** du menu général ou cliquez dans la barre de boutons sur le bouton .

Cette opération déclenche la création ou la mise à jour dans la base de données réelle, des trois tables contenant les éléments de redéfinition de la base.

Mise à jour d'une source de données virtuelle

Lorsque vous changez le nom d'un environnement, vous pouvez mettre à jour la source de données qui y correspond. Cette fonctionnalité est automatique si vous avez coché la case **Mise à jour automatique de la source de données** dans l'option **Affichage→Options** du menu général.



Si cette option est cochée, deux possibilités existent :

- Soit la case **Affichage des avertissements** de l'option **Affichage→Options** du menu général est cochée : dans ce cas, **Tun DB Revamp** vous préviendra à chaque fois que la mise à jour automatique va être effectuée pour vous permettre de confirmer ou non son exécution.
- Soit la case **Affichage des avertissements** de l'option **Affichage→Options** du menu général n'est pas cochée : dans ce cas, la mise à jour se fera automatiquement sans que vous la confirmiez.

En revanche, si l'option **Mise à jour automatique de la source de données** n'est pas cochée, utilisez l'option **Insertion→Mettre à jour la source de données associée** du menu général ou bien l'option **Mettre à jour la source de données associée** du menu contextuel relatif à l'environnement concerné pour mettre à jour la source de données liée à cet environnement.

Si vous souhaitez momentanément supprimer le lien entre un environnement et sa source de données, utilisez l'option **Insertion→Effacer la source de données associée** du menu général ou bien l'option **Effacer la source de données associée** du menu contextuel relatif à cet environnement. Recréez ensuite le lien en utilisant l'option **Mettre à jour la source de données associée**.

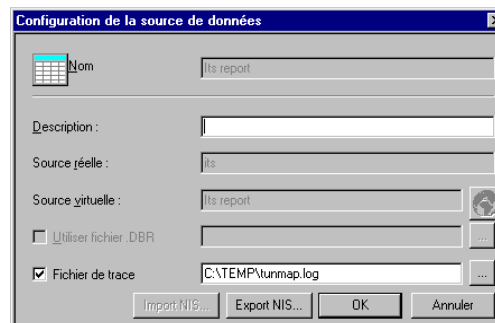


Création d'une source de données virtuelle

Dans le chapitre "**Configuration et utilisation**" est décrit comment créer une source de données virtuelle depuis l'**ODBC Administrator**.

Cette opération est également possible depuis **Tun DB Revamp**, en sélectionnant dans le menu contextuel relatif à cet environnement l'option **Créer la source de données associée....**

La boîte de dialogue suivante apparaît :

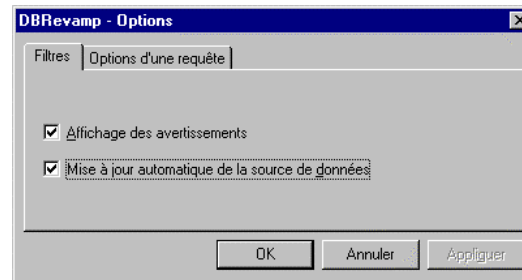


Entrez la description de votre choix pour cette source de données dans le champ **Description**. Sélectionnez la case à cocher **Fichier de trace** en indiquant le fichier voulu si vous souhaitez conserver les traces relatives à cette source de données virtuelle. Pour plus de précisions sur cette boîte de dialogue, reportez à "**Création d'une source de données virtuelles**" dans le chapitre "**Configuration et utilisation**".

Affichage des avertissements

Un certain nombre d'avertissements peuvent être donnés à l'administrateur au cours de l'utilisation de **Tun DB Revamp**. Par exemple, ces avertissements renseignent sur les incohérences générées au cours de la redéfinition de la base de données, ou bien préviennent des manques détectés dans la structure construite.


Par défaut, **Tun DB Revamp** affiche ces avertissements. Cependant, vous pouvez supprimer l'affichage de ces fenêtres en désélectionnant la case à cocher **Affichage des avertissements** de l'option **Affichage**→**Options** du menu général.



Gestion locale d'une source de données redéfinie

Vous pouvez sauvegarder et ouvrir localement la description de la base de données redéfinie. Ceci peut être utile si vous ne souhaitez pas exporter immédiatement la redéfinition de la base de données, ou pour en conserver des versions antérieures. Cette description est enregistrée dans un fichier dont l'extension est ".dbr".


➤ Sauvegarde locale

Pour sauvegarder localement la description d'une base de données redéfinie par **Tun DB Revamp**, sélectionnez l'option **Fichier**→**Enregistrer** du menu général (ou l'option **Fichier**→**Enregistrer sous** pour l'enregistrer sous un autre nom), ou bien cliquez dans la barre de boutons sur le bouton .

Le chemin d'accès à la source de données réelle est lui aussi sauvegardé, ce qui permet d'exporter ultérieurement la source de données et ses environnements sans avoir à renseigner sur sa destination.



➤ Ouverture d'une source de données locale

Pour ouvrir une source de données redéfinie enregistrée localement dans un fichier ".dbr", sélectionnez l'option **Fichier→Ouvrir** du menu général ou bien cliquez dans la barre de boutons sur le bouton .

Sélectionnez la source de données redéfinie de votre choix (fichier ".dbr").

Les dernières sources de données ouvertes précédemment sont disponibles depuis le menu **Fichier**.

➤ Rechargement de la structure d'une base de données




Lorsque vous ouvrez un fichier ".dbr" enregistré localement, vous pouvez mettre à jour la structure de la base de données réelle à partir de laquelle les environnements ont été créés. Ceci est très utile lorsque la base de données réelle a subi des modifications depuis le dernier enregistrement du fichier ".dbr". Pour cela, sélectionnez l'option **Fichier→Rechargement structure base de données** du menu général.



Il est recommandé à la suite de cette opération de mise à jour, de valider les environnements créés précédemment, notamment si des champs réels utilisés pour définir les champs virtuels ont été déplacés ou supprimés de la base. Voir "**Validation d'un environnement**" pour comprendre cette notion et savoir comment procéder.

Identification des champs

➤ Représentation des champs

Afin de faciliter la lecture du contenu d'une table, réelle ou virtuelle, **Tun DB Revamp** contient un certain nombre d'icônes pour représenter les champs.

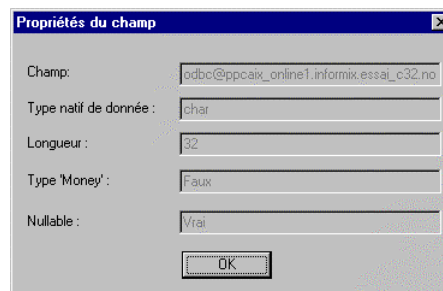
Icônes	Signification
	champ caractère
	champ date
	champ numérique

	champ binaire
	clé primaire de la table

➤ Propriétés des champs réels

Vous pouvez obtenir des informations complémentaires sur un champ d'une table réelle en sélectionnant l'option **Propriétés** du menu contextuel relatif à ce champ.

Une boîte de dialogue de ce type apparaît :



Propriétés du champ

Champ: jdbc@ppcaix_online1.informix.essai_c32.no

Type natif de donnée: char

Longueur: 32

Type 'Money': Faux

Nullable: Vrai

OK

Remarque

Le type du champ précisé ici correspond au type natif et dépend donc du SGBD utilisé.



PARTIE 3
ANNEXES

REFERENCES

Index

Remarque :

XXX représente l'extension relative au type de base de données :

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

CONFIG.XXX	Fichier contenant les paramètres d'exploitation et de sécurité du serveur UNIX de Tun SQL
DBMAP	Application Windows pour réaliser ou mettre à jour des tables de conversion de caractères.
DBSCRIPT	Application Windows pour interpréter et exécuter des "scripts" SQL
DBSHOW	Application Windows de test et de configuration
PARAM.XXX	Fichier contenant les paramètres d'installation du serveur UNIX de Tun SQL
TUNODBC200.XXX	Serveur UNIX de Tun SQL

CONFIG.XXX

Fichier contenant les paramètres d'exploitation et de sécurité du serveur de **Tun SQL**

➤ Description

Les fichiers **config.XXX** permettent de fournir un certain nombre de paramètres au serveur de **Tun SQL**. A la différence des fichiers **param.XXX**, les paramètres ne concernent pas le fonctionnement global du serveur mais sont définis base de données par base de données. A titre d'exemple, l'allure d'un fichier **config** est la suivante :

```
#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]
```

Ce fichier peut contenir autant de sections qu'il y a de base de données gérées par le SGBD associé au serveur UNIX de **Tun SQL**. Il n'est pas nécessaire d'y faire figurer toutes les base de données si vous n'avez pas de paramètres particuliers à affecter.



Chacune des sections est représentée par le nom de la base de données encadré par des crochets (par exemple **[tunsqldemo]**). Pour chacune des sections, les paramètres suivants peuvent être définis :

Define=ENV_VARIABLE :value

Permet d'affecter à une variable d'environnement **ENV_VARIABLE** une valeur **value** avant l'ouverture de la base de données (répertoire d'installation de la base de données, format de la date...). Il est possible de répéter cette option autant de fois que nécessaire dans le fichier **config**.

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

RowLimitValue=value

RowLimitMax=value

Certaines applications bureautiques laissent l'utilisateur libre de composer ses propres requêtes SQL. Par ailleurs, certaines applications ont tendance à lire le contenu de toute une table avant de l'afficher à l'écran. Cela ne pose pas de problèmes particuliers lorsqu'il s'agit de petites tables situées sur une base de données locale. Par contre, cela pose des problèmes insurmontables lorsqu'il s'agit de très grandes tables situées sur une base de données centralisée et distante. Dans ce dernier cas les échanges sur le réseau sont considérables et la mémoire du PC devient insuffisante pour stocker les données reçues. Très souvent le PC est obligé de "rebooter" après une telle requête "select".

Pour pallier cet inconvénient, le serveur UNIX de **Tun SQL** intègre une notion de limites accessible au travers du paramètre **RowLimitMode**. Si ce paramètre est défini, il est prioritaire par rapport aux valeurs qui peuvent être déclarées sur la source de données sur le PC.

Ce paramètre peut prendre 5 valeurs possibles :

None	Aucune limite n'est imposée par le driver ODBC
Absolute	Le driver ODBC refusera de charger plus de " RowLimitValue " rangées en une seule requête "select". Il n'affichera pas de messages pour en informer l'utilisateur.
Fixed	Le driver ODBC refusera de charger plus de " RowLimitValue " rangées en une seule requête "select". Il affichera un message pour en informer l'utilisateur.
Variable	Le driver ODBC refusera de charger plus de " RowLimitValue " rangées en une seule requête "select". Il affichera toutefois un message proposant d'en charger davantage dans la limite de la valeur supérieure (RowLimitMax)
Extended	Le driver ODBC refusera de charger plus de " RowLimitValue " rangées en une seule requête "select". Il affichera toutefois un message proposant d'en charger davantage sans limites supérieure. Dans ce dernier cas, le message affiché n'est finalement qu'un "Warning".

DbmsName=DatabaseName

Ce paramètre permet d'affecter ou de modifier le nom de la base de données qui sera transmis au driver ODBC.

Version=DatabaseVersion

Ce paramètre permet d'affecter ou de modifier le numéro de version de la base de données qui sera transmis au driver ODBC.



En plus des sections propres à chacune des bases de données, le fichier **config** peut comporter les sections **[Allowed]** et **[Denied]** qui permettent de sécuriser l'accès à certaines bases de données. Ces deux sections fonctionnent de la manière suivante :

[Allowed]

Cette section doit contenir une succession de triplets de la forme :

base_name , user_name , product_name

où

- **base_name** est le nom d'une base de donnée
- **user_name** est le nom d'un utilisateur de la base de données
- **product_name** est le nom d'une application Windows qui va exploiter le serveur.

Chaque triplet permet d'indiquer si un utilisateur (**user_name**) est habilité à utiliser la base de données (**base_name**) avec l'application Windows (**product_name**). Chaque paramètre peut être remplacé par le caractère générique '*'.

Par exemple, le triplet **tunsqldemo,* , excel** signifie que tous les utilisateurs peuvent utiliser la base **tunsqldemo** à l'aide de l'application **Excel** sauf ceux qui pourraient figurer dans un triplet identique situé sous la section **[Denied]**.

[Denied]

Cette section doit contenir une succession de triplets de la même forme que dans la section **[Allowed]**.

Chaque triplet permet d'indiquer qu'un utilisateur (**user_name**) n'est pas habilité à utiliser la base de données (**base_name**) avec l'application Windows (**product_name**). Chaque paramètre peut être remplacé par le caractère générique "*".

Par exemple, le triplet ***, jean, excel** signifie que l'utilisateur **jean** ne peut utiliser aucune base de données à l'aide de l'application **Excel** sauf celles qui pourraient figurer dans un triplet identique et contradictoire situé sous la section **[Allowed]**.

Remarque

Pour qu'un serveur UNIX de **Tun SQL** prenne en compte un fichier **config**, il est nécessaire de lui indiquer à l'aide de l'option **-c** sur la ligne de commandes.

Sous Informix peuvent coexister deux moteurs de bases de données différents (Informix version 5 et Informix version 7). Par exemple, une première base est accédée par le moteur 1 installé dans le répertoire **/u/informix1**, une deuxième base est accédée par le moteur 2 installé dans le répertoire **/u/informix2**. Dans ce cas, le fichier **config.ifx** aura pour contenu :

```
[base1]
Define=INFORMIXDIR :/u/informix1
Define=DBPATH :/u/database1
Version=5.01
[base2]
Define=INFORMIXDIR :/u/informix2
Define=DBPATH :/u/database2
Version=7.01
```

➤ **Voir aussi**

param.XXX, tunodbc200.XXX



DBMAP

Application Windows pour créer ou mettre à jour des tables de conversion

➤ Syntaxe

```
DBMAP [-ffile_name]
```

➤ Description

Tun DB Map permet de créer ou de mettre à jour des tables de translation. Les tables de translations permettent d'affranchir l'utilisateur des différences de notation des caractères nationaux pouvant exister entre le PC et le SGBD distant. Pour pouvoir être prise en compte, les tables de translation doivent être déclarées dans la définition de la source de données utilisée.

-ffile_name

Permet d'indiquer le nom du fichier contenant une table de translation existante.

DBSCRIPT

Application Windows pour exécuter des scripts SQL.

➤ Syntaxe

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

➤ Description

Tun DB Script permet d'exécuter en une seule opération tout un train de requêtes SQL. Il interprète les ordres SQL les uns après les autres et s'arrête dès qu'il rencontre une erreur. **Tun DB Script** permet aussi de modifier et de sauvegarder des batches SQL.

Tun DB Script est utile pour télécharger le contenu d'une base de données à partir d'un PC Windows sur un SGBD distant. Il permet aussi d'effectuer des mises à jour massives ou des purges régulières sur le contenu de la base de données.

-ddata_source

Permet d'indiquer le nom de la source de données sur laquelle sera exécuté le batch SQL. **Tun DB Script** ne réalise pas la connexion à cette source de données. Il faut le faire ensuite à la main.

-ffile_name

Permet d'indiquer le nom du fichier contenant les ordres SQL qui sera chargé lors du démarrage de l'application.



DBSHOW

Application Windows de test et de configuration.

➤ Syntaxe

```
DBSHOW [-hhost_name]
```

➤ Description

Tun DB Show permet d'interroger un serveur du réseau pour vérifier si un ou plusieurs serveurs de **Tun SQL** y sont installés.

Il suffit d'entrer le nom du serveur dans le champ **Serveur** puis de cliquer sur le bouton **Rechercher serveurs...** pour obtenir cette information.

Si un ou plusieurs serveurs de **Tun SQL** sont convenablement installés sur la machine distante, alors **Tun DB Show** retourne leur nom ainsi que le nom du SGBD avec lequel ils sont interfacés.

Cette application est particulièrement utile pour vérifier la conformité de l'installation.

-hhost_name

Permet d'indiquer le nom du serveur sur lequel l'interrogation devra être effectuée.

PARAM.XXX

Fichier contenant les paramètres de lancement du serveur UNIX de **Tun SQL**

➤ Description

Plutôt que de lancer les serveurs de **Tun SQL** avec de très nombreuses options sur la ligne de commande, il est préférable de ranger toutes les options les unes sous les autres dans un fichier et de passer ce fichier au serveur derrière l'option **-f**.

La procédure d'installation de **Tun SQL** utilise ce mécanisme et range les options dans des fichiers **param.XXX** où XXX est remplacé par le nom du SGBD sur trois lettres (param.ora, param.syb, param.ifx, param.db2, param.pro, ...).

Voici à titre d'exemple le contenu d'un tel fichier :

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

La signification des différentes options est commentée dans la section **TUNODBC200.XXX**

➤ Spécificités Progress

Certains champs Progress peuvent contenir plusieurs valeurs : ce sont des champs-tableaux (array fields). Pour visualiser ces différentes valeurs dans des applications de type MS-Query ou MS-Access, l'option suivante doit être présente dans le fichier param.proX (où X est le numéro de la version Progress utilisée) :

```
-arrayfields=*
```

où * est un des caractères suivants :

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



Par défaut, le caractère utilisé est _.

Les colonnes nécessaires à la visualisation des différentes valeurs du champ-tableau seront alors nommées :

columnname*n*,

où * est le caractère choisi dans le fichier param.proX (par défaut _) et n la position de la valeur dans le tableau..

Exemple :

La deuxième valeur du champ tableau apparaîtra dans la colonne col_2_.

Si l'utilisation du caractère _ pose un problème pour générer la colonne (d'autres colonnes portent un nom similaire), vous devez alors choisir un autre caractère parmi les cinq autres possibles.

➤ **Voir aussi**

config.XXX, tunodbc200.XXX

TUNODBC200.XXX

Serveur de **Tun SQL**.

Remarque

Les différents serveurs de **Tun SQL** ont un certain nombre d'options en commun. La liste des différentes options est obtenue en invoquant l'exécutable **tunodbc200.xxx** avec l'option **-a[ll]**, xxx représentant l'extension relative au type de base de données.

Dans la liste de toutes les options possibles obtenues par cette commande, celles relatives à un type de bases de données particulier sont soulignées.

➤ Syntaxe

```
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...
```



➤ Description des options communes

-a

Permet de lister toutes les options supportées par **tunodbc200.XXX**.

-c=config_file

Permet d'associer un fichier de configuration (**config.XXX**) au serveur (Cf. **config.XXX**).

-db=name

Permet d'associer un nom de SGBD au serveur de **Tun SQL**. C'est la valeur qui sera affichée dans la liste lors de l'exécution de **Tun DB Show**.

-de

Indique que le serveur doit fonctionner en mode trace. Les messages seront affichés par défaut sur le device **/dev/console**.

-Dname=value

Permet d'affecter à une variable d'environnement **name** une valeur **value** avant l'exécution du serveur (répertoire d'installation de la base de données, format de la date...). Il est possible de répéter cette option autant de fois que nécessaire dans la ligne de commande. Il est absolument nécessaire de définir certaines variables pour que le serveur de **Tun SQL** puisse fonctionner avec certains SGBD. Cette définition est en réalisée par la procédure d'installation. Pour mémoire, ces variables sont nécessaires pour les SGBD suivants :

Oracle :

ORACLE_HOME : Répertoire d'installation d'Oracle
ORACLE_SID : Base de données par défaut

Informix :

INFORMIX_DIR : Répertoire d'installation d'Informix
DBPATH : Répertoire où se trouve la base de données (SE uniquement)

Sybase :

SYBASE : Répertoire d'installation de Sybase

SYBSERVNAME : Identificateur du fichier de paramètres du serveur (optionnel). Equivalent à la variable DSQUERY définie et utilisée par SYBASE.

-f=param_file

Permet d'indiquer le nom d'un fichier dans lequel sont regroupées les unes derrière les autres toutes les options décrites pour ce programme. Cette option permet de lancer le programme sans obligatoirement le faire suivre de dizaines d'options différentes.

-i

Permet d'entrer en mode interactif de tests pour contrôler le bon fonctionnement du serveur.

-o=file

Permet d'indiquer le nom du fichier ou du device sur lequel seront écrits les messages de trace du serveur et de son contrôleur d'accès (Watchdog). Pour le mode **debug** seulement.

-o2=file

Permet d'indiquer le nom du fichier ou du device sur lequel seront écrits les messages de trace du WatchDog, séparément des messages de trace du serveur. Pour le mode **debug** seulement.

-t=xx

Permet d'affecter une valeur de timeout. C'est le temps au bout duquel toute absence de réponse de la part du PC sera considérée comme un arrêt du PC. Le serveur UNIX de **Tun SQL** s'arrêtera donc lui aussi de fonctionner. Cette valeur est moins prioritaire que celle qui est affectée, coté PC, lors de la définition de la source de données.

-u

Permet de définir la liste d'utilisateurs autorisés ("*" pour tous). La valeur par défaut est *. Par exemple :

x = *

u = toto (seul "toto" est autorisé)



-v=XX

Permet d'associer un numéro de version de SGBD au serveur de **Tun SQL**. C'est une valeur qui sera affichée dans la liste lors de l'exécution de **Tun DB Show**.

-x

Permet de définir la liste d'utilisateurs interdits ("*" pour tous). Par exemple :

```
u = *  
x = toto                (seul "toto" est interdit)
```

➤ **Description des options particulières à Informix**

-h

Par défaut, Informix ne conserve pas les curseurs ouverts lorsqu'on exécute une commande **commit** ou **rollback**. Si les applications utilisant ce serveur ne savent pas gérer ce cas, cette option permet de conserver les curseurs ouverts après exécution d'une telle commande.

-n

Uniquement sur SCO-UNIX v5. Si le système SCO-UNIX 3.2 version 5 n'arrive pas à contrôler correctement les mots de passe des utilisateurs, cette option permet de supprimer le contrôle des mots de passe.

-s

Par défaut, Informix ne sait pas effectuer des ordres **select** avec tri (**group by** ou **order by**) sur une colonne qui n'a pas été sélectionnée dans l'ordre **select**. Si les applications utilisant ce serveur ne savent pas gérer ce cas, cette option permet de compenser ce manque.

➤ Description des options particulières à Oracle

-l

Si dans le catalogue des tables, des colonnes, des index ou des vues ont été créés avec des caractères minuscules, cette option autorise les fonctions du catalogue à retourner toutes les informations entourées d'un caractère guillemet ("). Les applications utilisant ce serveur créeront leurs requêtes avec ces noms entourés de guillemets.

➤ Description des options particulières à Progress

-n

Uniquement sur SCO-UNIX v5. Si le système SCO-UNIX 3.2 version 5 n'arrive pas à contrôler correctement les mots de passe des utilisateurs, cette option permet de supprimer le contrôle des mots de passe.

-nor

La base de données Progress ne sait pas combien de lignes ont été modifiées ou supprimées lors d'un ordre **update** ou **delete**. Par défaut, ce serveur compense ce manque. Cependant, cela coûte du temps à chaque ordre **update** ou **delete** donné. Si les applications utilisant ce serveur ne nécessitent pas d'avoir le nombre de lignes modifiées, cette option permet de supprimer la compensation du serveur et de gagner du temps.

-ow

Par défaut, le serveur ne supporte pas la notion de propriétaire (owner) d'objets dans la base. En effet, si les propriétaires sont retournés avec les objets, certaines applications tentent de préfixer ces objets par le nom du propriétaire, ce qui entraîne des erreurs lors de l'exécution. Si l'application a besoin de connaître le propriétaire, et que la gestion de celui-ci est correctement effectuée, cette option peut être utilisée.



-p=XX

Si des options particulières de Progress doivent être utilisées (par exemple l'option -Q qui force la base de données à suivre la norme ANSI), cette option doit être utilisée en remplaçant XX par une chaîne entre guillemets contenant toutes les options souhaitées.

-sv

La base de données Progress n'a qu'un seul type de chaîne de caractères. Par défaut, toutes ces chaînes sont perçues comme étant de longueur fixe (**SQL_CHAR** dans ODBC). Si cette options est utilisée, elles seront considérées comme des chaînes de longueur variable (**SQL_VARCHAR** dans ODBC).

-sy

La base de données Progress peut définir des colonnes systèmes ou cachées, qui ne seront pas retournées si l'on effectue une recherche de toutes les colonnes de façon non nominative (**select * from**). C'est pourquoi par défaut ce serveur ne décrit pas ces colonnes dans son catalogue. Si malgré tout, ces colonnes cachées sont nécessaires, cette option inhibe leur non retour.

➤ **Voir aussi**

param.XXX, config.XXX

ORDRES SQL POUR C-ISAM

Liste des principaux ordres

CREATE DATABASE	107
CREATE TABLE	108
DEFINE TABLE.....	109
COLUMN DEFINITION OPTION.....	110
DEFAULT CLAUSE	111
NOT NULL CLAUSE.....	112
CONSTRAINT DEFINITION SUBSET.....	113
CONSTRAINT DEFINITION OPTION.....	114
FILE IS OPTION	115
CREATE INDEX.....	116
CREATE SYNONYM	117
COMMENT	118
DROP DATABASE.....	119
CONNECT DATABASE.....	120
DISCONNECT DATABASE	121
DROP INDEX.....	122
DROP TABLE	123
DROP SYNONYM.....	124

UNDEFINE TABLE	125
SELECT	126
SELECT CLAUSE	127
EXPRESSION	128
FROM CLAUSE	129
WHERE CLAUSE	130
GROUP BY CLAUSE.....	131
HAVING CLAUSE	132
ORDER BY CLAUSE.....	133
DELETE.....	134
INSERT.....	135
VALUES CLAUSE.....	136
UPDATE.....	137
SET CLAUSE.....	138
AGGREGATE EXPRESSION.....	139

Syntaxe des ordres SQL

Les ordres SQL sont présentés selon la syntaxe suivante :

- Les noms réservés sont en majuscules (INSERT, UNIQUE...). Ils peuvent cependant être entrés en minuscules sur la ligne de commande.
- Les noms de variables sont en italique (*Databasename...*).
- Une séquence entre crochets est optionnelle ([optionnel]).
- Les accolades et le terme xor qualifient un choix exclusif ({A xor B}).
- Le n exposant (ⁿ) qualifie une séquence qui peut être répétée de 0 à n fois (séquenceⁿ)

Les signes de ponctuation et les parenthèses sont des symboles littéraux qui doivent être tapés exactement comme indiqué.



CREATE DATABASE

➤ Objectif

Crée une nouvelle base de données.

➤ Syntaxe

```
CREATE DATABASE Basename
```

Remarque

Le nom de la base de données doit être inférieur à 18 caractères.

➤ Usage

La base de données créée devient la base de données courante.

Cet ordre ne peut être utilisé que par l'outil **sqltools** (outil **Tun SQL**).

Après la création de la base de données, un répertoire nommé *basename.ism* est créé. Ce répertoire contient les fichiers C-ISAM supplémentaires formant le catalogue de la base de données (SysTables, SysColumns, SysIndexes, SysDefaults). C'est un sous-répertoire du répertoire désigné par la variable d'environnement ISAM-PATH si elle est renseignée, sinon du répertoire courant.

➤ Exemple

```
CREATE DATABASE TEST;
```

Crée le répertoire *test.ism* qui contient les fichiers SysTables.dat, SysTables.idx, SysColumns.dat, SysColumns.idx, SysIndexes.dat, SysIndexes.idx, SysDefaults.dat, et SysDefaults.idx.

CREATE TABLE

➤ Objectif

Crée une nouvelle table dans la base de données courante et fixe les contraintes d'intégrité des données sur les colonnes (ou groupe de colonnes).

➤ Syntaxe

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

Remarque

Le nom de la table doit être inférieur à 18 caractères.

➤ Usage

Le nom d'une table dans une base de données doit être unique. Il en est de même pour les colonnes d'une même table.

Le nom d'une table peut être précédé du nom d'un utilisateur UNIX qui devient alors le propriétaire de la table. Si aucun nom n'est précisé, le login courant est utilisé par défaut.

➤ Exemple

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

Cette instruction a pour effet de créer la table table1 dans la base de données relationnelle en créant les fichiers C-ISAM associés (table1_100.dat et table1_100.idx, par exemple).

Le nom de fichier C-ISAM est créé en prenant les 7 premiers caractères du nom de la table et en ajoutant une valeur unique. Si la table a un nom de moins de 7 caractères, il est complété par des "_". La valeur unique est 100 pour la première table créée, puis est incrémentée de 1 à chaque table créée.



DEFINE TABLE

➤ Objectif

Définit une nouvelle table dans la base de données courante, avec contraintes d'intégrité des données sur ses colonnes ou une combinaison de ses colonnes, et options de condition d'existence des fichiers.

➤ Syntaxe

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

Remarque

Le nom de la table doit être inférieur à 18 caractères.

➤ Usage

Le nom d'une table dans une base de données doit être unique. Il en est de même pour les colonnes d'une même table.

Le nom d'une table peut être précédé du nom d'un utilisateur UNIX qui devient alors le propriétaire de la table. Si aucun nom n'est précisé, le login courant est utilisé par défaut.

➤ Exemple

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

Dans cet exemple les fichiers C-ISAM file_1 (file_1.idx et file_1.dat) existent déjà, il faut juste définir la table associée.

COLUMN DEFINITION OPTION

Utilisé dans les ordres **CREATE TABLE** et **DEFINE TABLE**.

➤ Objectif

L'option "column definition" dans l'ordre **DEFINE TABLE** (ou l'ordre **CREATE TABLE**) permet de lister le nom, le type, la valeur par défaut et la contrainte d'une seule colonne.

➤ Syntaxe

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

➤ Exemple

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

Cette table est composée des colonnes *name*, *race* et *sex*.



DEFAULT CLAUSE

Utilisé dans l'option **COLUMN DEFINITION**.

➤ Syntaxe

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

➤ Usage

La valeur par défaut est insérée dans la colonne quand aucune valeur explicite n'est spécifiée. Si aucune valeur par défaut n'est précisée, et que la colonne accepte une valeur nulle, la valeur par défaut est NULL.

LITERAL	représente une chaîne de caractères ou un caractère numérique constant défini par l'utilisateur
NULL	représente la valeur nulle
CURRENT	représente la date et l'heure courantes (seulement utilisable avec un type TIMESTAMP)
TODAY	représente la date courante (seulement utilisable avec le type DATE)
USER	représente le nom de l'utilisateur courant (seulement utilisable avec les types VAR et VARCHAR)

➤ Exemple

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

Dans cette table, la valeur par défaut de *sex* est une valeur littérale : 'M'.

Utilisé dans l'option **COLUMN DEFINITION**.

➤ Objectif

Si vous n'indiquez pas de valeur par défaut dans une colonne, la valeur par défaut est "null" sauf si vous précisez les mots clés NOT NULL après le type de données de la colonne. Dans ce cas, il n'y a pas de valeur par défaut pour la colonne.

➤ Syntaxe

NOT NULL

➤ Exemple

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

Si la colonne est désignée comme NOT NULL (et qu'aucune valeur par défaut n'est précisée), vous devez entrer une valeur dans cette colonne lors de l'insertion d'une ligne ou lors de la mise à jour de cette colonne. Sinon, le serveur retourne une erreur.



CONSTRAINT DEFINITION SUBSET

Utilisé dans l'option **COLUMN DEFINITION**.

➤ Objectif

Le sous-ensemble "constraint definition" permet de créer des contraintes pour une simple colonne.

➤ Syntaxe

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

Remarque :

Le nom de la contrainte ne doit pas dépasser plus de 18 caractères et doit être unique dans toute la base de données.

➤ Usage

UNIQUE	contraint le champ à l'unicité
PRIMARY KEY	contraint le champ à être unique et à être la clé primaire de la table

Si le nom de la contrainte n'est pas précisé, un nom par défaut sera attribué.

➤ Exemple

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

La contrainte d'unicité du numéro de facture est appelée *un_invoice*.

CONSTRAINT DEFINITION OPTION

Utilisé dans les ordres **CREATE TABLE** et **DEFINE TABLE**.

➤ **Objet**

L'option "constraint definition" permet de créer des contraintes pour un ensemble de colonnes (de 1 à 8 colonnes).

➤ **Syntaxe**

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

➤ **Usage**

UNIQUE	contraint le champ à l'unicité pour l'ensemble des colonnes
PRIMARY KEY	contraint le champ à être unique et à être la clé primaire pour l'ensemble des colonnes

Si le nom de la contrainte n'est pas précisé, un nom par défaut sera attribué.

Chaque colonne nommée dans la contrainte doit être une colonne de la table et ne peut pas apparaître dans la liste plus d'une fois.

➤ **Exemple**

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

La contrainte de clé primaire *pk_family* porte sur les champs *name* et *surname*.



Utilisé dans l'ordre **DEFINE TABLE**.

➤ Objectif

Définit l'usage de la table conformément à la présence ou l'absence d'un fichier.

➤ Syntaxe

FILE IS *filename*

➤ Usage

Si cette option est utilisée, aucun fichier C-ISAM supplémentaire n'est créé. **Tun SQL** doit trouver alors les fichiers *filename.dat* et *filename.idx* dans le répertoire de la base de données courant pour pouvoir exploiter la table.

Si cette option n'est pas utilisée, un nom par défaut est attribué par défaut aux fichiers créés dans le répertoire de la base de données courante.

➤ Exemples

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

Dans cet exemple, aucun fichier n'est créé. Les fichiers *fic1.dat* et *fic1.idx* doivent se trouver dans le répertoire *test.ism*.

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

Dans cet exemple, les fichiers *table1_100.dat* et *table1_100.idx* (noms par défaut) sont créés dans le répertoire *test.ism*.

CREATE INDEX

➤ Objectif

Crée un index pour une ou plusieurs colonnes dans la table (de 1 à 8 colonnes).

➤ Syntaxe

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

Remarques :

Le nom de l'index ne doit pas dépasser 18 caractères.
Le nombre de colonnes peut varier de 1 à 8.

➤ Usage

UNIQUE	contraint l'index à être unique
DISTINCT	synonyme pour UNIQUE

Pour les tables définies avec l'option FILE IS, l'ordre CREATE INDEX doit être exécuté avant la copie des fichiers filename.dat et filename.idx.

➤ Exemple

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

Cette instruction crée l'index *ix_name* sur les colonnes *name* et *birth_date* de la table *Table1*.



CREATE SYNONYM

➤ Objectif

Attribue un synonyme à une table.

➤ Syntaxe

```
CREATE SYNONYM synonymname FOR tablename
```

Remarques :

Le nom du synonyme ne doit pas dépasser 18 caractères.
--

➤ Usage

Le nom d'un synonyme peut être précédé du nom d'un utilisateur UNIX qui devient alors le propriétaire du synonyme. Si aucun nom n'est précisé, le login courant est utilisé par défaut.

➤ Exemple

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

Cette instruction crée le synonyme *employee* pour la table *Table1*.

COMMENT

➤ Objectif

Attribue un commentaire à une table ou un synonyme.

➤ Syntaxe

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

➤ Exemple

```
COMMENT on TABLE1 IS 'Table des employés'
```



DROP DATABASE

➤ Objectif

Supprime une base de données complète, y compris tous les catalogues, index et données.

➤ Syntaxe

DROP DATABASE *basename*

Remarque :

Le nom de la base de données ne doit pas dépasser 18 caractères.

➤ Usage

Une base de données qui est utilisée par un autre utilisateur ne peut pas être supprimée.

L'ordre DROP DATABASE ne supprime pas le répertoire de la base de données s'il inclut d'autres fichiers que ceux créés pour les tables et index de la base supprimée.

➤ Exemple

```
DROP DATABASE DBTEST;
```

Supprime la base de données *DBTEST*.

CONNECT DATABASE

➤ Objectif

Etablit la connexion avec une autre base de données. On ne peut pas effectuer d'opérations sur une base de données si on n'y est pas connecté.

➤ Syntaxe

```
CONNECT DATABASE basename
```

➤ Exemple

```
CONNECT DATABASE DBTEST2;
```

Etablit la connexion avec la base *DBTEST2*.



DISCONNECT DATABASE

➤ Objectif

Ferme la connexion avec la base de données courante.

➤ Syntaxe

DISCONNECT DATABASE *basename*

➤ Exemple

```
DISCONNECT DATABASE DBTEST2;
```

Rompt la connexion avec la base *DBTEST2*.

DROP INDEX

➤ Objectif

Supprime l'index.

➤ Syntaxe

DROP INDEX *indexname*

➤ Exemple

```
DROP INDEX ix_name ;
```

Supprime l'index *ix_name*.



DROP TABLE

➤ Objectif

Supprime une table, ainsi que ces index et données associés.

➤ Syntaxe

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

➤ Usage

Si un synonyme est supprimé par l'ordre **DROP TABLE**, alors la table et son synonyme sont supprimés.

Si l'ordre **DROP TABLE** s'applique à une table, alors les synonymes de cette table ne seront supprimés que sur un ordre **DROP SYNONYM**.

➤ Exemple

```
DROP TABLE TABLE1;
```

Supprime la table TABLE1, ses index et ses données.

DROP SYNONYM

➤ Objectif

Supprime un synonyme de table préalablement défini.

➤ Syntaxe

```
DROP SYNONYM synonymname
```

➤ Usage

Si une table est supprimée, le synonyme reste en place jusqu'à ce qu'il soit explicitement supprimé avec l'ordre DROP SYNONYM.

➤ Exemple

```
DROP SYNONYM employee;
```

Cette instruction supprime le synonyme *employee* attribué à la table *Table1*. Celle-ci n'est pas supprimée sur cet ordre.



UNDEFINE TABLE

➤ Objectif

Supprime une table définie par un ordre DEFINE, sans supprimer les fichiers données et index associés.

➤ Syntaxe

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

➤ Exemple

```
UNDEFINE TABLE TABLE1;
```

Supprime la table TABLE1 créée par DEFINE TABLE TABLE1.

SELECT

➤ Objectif

Interroge une base de données.

➤ Syntaxe

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

➤ Usage

Vous pouvez interroger les tables de la base de données courante ou de toute autre base de données.

➤ Exemple

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

Cette requête va chercher dans la table *customers* l'ensemble des clients ayant un chiffre d'affaires supérieur à 250 et donne leurs noms, classés par pays.



SELECT CLAUSE

Utilisé par les ordres **SELECT** et **INSERT**.

➤ Syntaxe

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

Dans la clause **SELECT**, précisez exactement quelle donnée est sélectionnée, et précisez également si vous souhaitez omettre les doublons ou non.

➤ Usage

ALL	précise si toutes les valeurs sélectionnées sont retournées, même si elles sont dupliquées
DISTINCT	élimine les rangées dupliquées
UNIQUE	synonyme pour DISTINCT

➤ Exemple

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

Cette requête va chercher dans la table *customers* l'ensemble des clients ayant un chiffre d'affaires supérieur à 250 et donne leurs noms, classés par pays.

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

Cette requête rend la date de commande, le nombre de commandes et la différence entre les dates de paiement et de commande groupées par date de commande et différence de dates.

Utilisé par la clause **SELECT**.

➤ Syntaxe

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

➤ Exemple

```
'Cordwainer'
```

La chaîne de caractère '*Cordwainer*' est une sous-expression.



FROM CLAUSE

Utilisé par l'ordre **SELECT** et **DELETE**.

➤ Objectif

Liste la ou les table(s) dont sont sélectionnées les données.

➤ Syntaxe

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

➤ Exemple

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

Dans cet exemple les données sont extraites des tables **customers** et **orders** auxquelles on assigne des alias.

WHERE CLAUSE

Utilisé par les ordres **SELECT**, **DELETE** et **UPDATE**.

➤ Objectif

Précise les critères de recherche et les condition de joint sur les données sélectionnées.

➤ Syntaxe

WHERE Condition [AND Condition]¹

➤ Exemple

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

Dans cet exemple, le critère de recherche porte sur la date de dernière commande.



GROUP BY CLAUSE

Utilisé par l'ordre **SELECT**.

➤ Objectif

Produit une simple rangée de résultats pour chaque groupe.

➤ Syntaxe

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

➤ Usage

Un groupe est un ensemble de colonnes qui ont les mêmes valeurs pour chaque colonne listée.

La variable "select number" est un entier représentant la position d'une colonne dans la clause SELECT.

➤ Exemple

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

Les résultats sont groupés par **order_date** et par **paid_date - order_date**.

HAVING CLAUSE

Utilisé par l'ordre **SELECT**.

➤ Objectif

Applique une ou plusieurs conditions à des groupes.

➤ Syntaxe

HAVING Condition

➤ Exemple

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

Cette requête retourne les call_code, call_dtime et customer_num et les groupe par call_code pour tous les appels venant de clients dont le customer_num est inférieur à 42.



ORDER BY CLAUSE

Utilisé par l'ordre **SELECT**.

➤ Objectif

Classe les résultats d'une requête dans l'ordre des valeurs contenues dans une ou plusieurs colonnes.

➤ Syntaxe

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

➤ Usage

La variable "select number" est un entier représentant la position d'une colonne dans la clause **SELECT**.

➤ Exemple

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

Cette requête donne les résultats classés par pays.

DELETE

➤ Objectif

Détruit une ou plusieurs lignes d'une table.

➤ Syntaxe

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

➤ Exemple

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

Cette instructions détruit les lignes de la table *customers* où la date de dernière commande est inférieure à 1992.



INSERT

➤ Objectif

Insère une ou plusieurs lignes dans une table.

➤ Syntaxe

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

➤ Exemple

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

Cette instruction insère les valeurs *'Socks'*, *'Cat'* et *'M'* dans la table *Pets*.

Utilisé par l'ordre **INSERT**.

➤ **Syntaxe**

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

➤ **Usage**

When you use the VALUES clause, you can insert only one row at a time. Each value that follows the VALUES keyword is assigned to the corresponding column listed in the INSERT INTO clause (or in column order if a list of columns is not specified).

➤ **Exemple**

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

Cette instruction insère les valeurs '*Socks*' dans la première colonne de *Pets*, '*M*' dans la troisième. La deuxième colonne n'a pas été renseignée.



UPDATE

➤ Objectif

Change la valeur de une ou plusieurs colonnes d'une ou plusieurs lignes dans une table.

➤ Syntaxe

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

➤ Exemple

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

Cette instruction passe à 10 le prix de tous les articles de type '*L*' dans la table *Catalog*.

SET CLAUSE

Utilisé par l'ordre **UPDATE**.

➤ Syntaxe

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

➤ Exemple

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

Dans cette instruction, la clause **SET** passe le prix `item_price` à 10.



AGGREGATE EXPRESSION

Utilisé par l'ordre **SELECT** ou dans une expression.

➤ Syntaxe

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

➤ Exemple

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

Cette requête retourne le nombre de types d'article différents dans la table *Catalog*.

Types de données

Cette section décrit les types de données supportés dans le langage SQL avec leurs définitions équivalentes dans le langage C. Ces descriptions correspondent aux types SQL utilisés au sein de l'ordre **CREATE TABLE**. Il s'agit là des correspondances en mode natif. Pour l'importation de fichiers créés hors de la database C-ISAM grâce à l'ordre **DEFINE TABLE**, quelques différences sont notifiées.

Dans le tableau suivant, les champs en italique dans la colonne intitulée "Type langage C" n'existent que dans le cas de l'utilisation d'une création de table.

Type SQL dans la base	Type SQL dans ODBC	Type langage C
bit	SQL_BIT	<i>unsigned char notnull_data;</i> unsigned char data;
byte	SQL_TINYINT	<i>unsigned char notnull_data;</i> unsigned char data;
char(maxlength) 1 <= maxlength <= 32511	SQL_CHAR	char data[maxlength];

varchar(maxlength) 1 <= maxlength <= 32511	SQL_VARCHAR	char data[maxlength];
binary(maxlength) 1 <= maxlength <= 32511	SQL_BINARY	<i>unsigned char notnull_data;</i> unsigned char data[maxlength];
varbinary(maxlength) 1 <= maxlength <= 32511	SQL_VARBINARY	<i>unsigned char size_data[2];</i> <i>unsigned char data[maxlength];</i>
smallint	SQL_SMALLINT	short data; /* 2 octets */
longint	SQL_INTEGER	long data; /* 4 octets */
real	SQL_FLOAT	float data; /* 4 octets */
double	SQL_DOUBLE	double data; /* 8 octets */
decimal(prectot, precdec) 1<=prectot<=32 et 0<=precdec<=prectot	SQL_DECIMAL	char data[(prectot+1)/2+1];
date	SQL_DATE	unsigned long data;
time	SQL_TIME	unsigned long data;
timestamp	SQL_TIMESTAMP	unsigned long data;

➤ Le type bit

Ce type correspond au type SQL_BIT dans ODBC. Il permet de stocker les valeurs binaires 0, 1 ou la valeur nulle (null).

Dans le cas de CREATE TABLE lors de l'utilisation de ce type de données, une zone de deux octets est réservée dans le fichier créé, afin de pouvoir différencier la valeur nulle de 0 et 1. Si un tel champ est utilisé dans la définition d'une clef, les deux octets seront utilisés dans la clef.

Dans le cas de DEFINE TABLE lors de l'utilisation de ce type de données, un seul octet est réservé. Ceci interdit donc la différenciation de la valeur 0 de la valeur nulle. C'est pourquoi dans DEFINE TABLE ce type de champ est nécessairement non nullable.

Le tableau suivant donne les valeurs stockées. Les données en italique n'ont de sens que dans le cas de CREATE TABLE.

champ de type bit	<i>notnull_data</i>	valeur de data
0	<i>1</i>	0
1	<i>1</i>	1
<i>null</i>	<i>0</i>	0

➤ Le type byte

Ce type correspond au type SQL_TINYINT dans ODBC. Il permet de stocker les valeurs de 0 à 255 ou la valeur nulle.



Dans le cas de CREATE TABLE lors de l'utilisation de ce type de données, une zone de deux octets est réservée comme pour le type **bit** dans le but de différencier la valeur nulle des autres valeurs. Si un tel champ est utilisé dans la définition d'une clef, les deux octets seront utilisés dans la clef.

Dans le cas de DEFINE TABLE lors de l'utilisation de ce type de données, un seul octet est réservé. Ceci interdit donc la différenciation de la valeur 0 de la valeur nulle. C'est pourquoi dans DEFINE TABLE ce type de champ est nécessairement non nullable.

Le tableau suivant donne les valeurs stockées. Les données en italique n'ont de sens que dans le cas de CREATE TABLE.

champ de type byte	<i>nonnull_data</i>	valeur de data
0	<i>1</i>	0
1	<i>1</i>	1
...	<i>1</i>	...
255	<i>1</i>	255
<i>null</i>	<i>0</i>	<i>0</i>

➤ Le type char

Ce type correspond au type SQL_CHAR dans ODBC. Il permet de stocker de 1 à 32511 caractères.

Lors de l'insertion d'une donnée dans un champ de ce type, tous les espaces non significatifs en fin de chaîne sont supprimés et la zone correspondante dans le fichier est comblée par des caractères "\0" (code ascii 0). Lors de la lecture d'une donnée stockée dans ces champs, la chaîne lue est automatiquement comblée à la taille maximum par des espaces (code ascii 32). Si une chaîne vide est stockée dans un champ de type **char**, un caractère espace unique est écrit, afin de différencier la chaîne vide de la valeur nulle.

champ de type char(10)	valeur de data
''	0x32000000000000000000
' '	0x32000000000000000000
'A '	0x41000000000000000000
'AaBb'	0x41614262000000000000
null	0x00000000000000000000

➤ Le type varchar

Ce type correspond au type SQL_VARCHAR dans ODBC. Le fonctionnement de ce type est pratiquement le même que celui du type **char**. Il permet de la même façon de stocker de 1 à 32511 caractères.

La différence avec le type **char** est que lors de l'insertion d'une donnée dans un champ de ce type, tous les espaces non significatifs en fin de chaîne ne sont pas supprimés, mais la zone correspondante dans le fichier est comblée par des caractères '\0' (code ascii 0).

Lors de la lecture d'une donnée stockée dans ces champs, la chaîne récupérée dans ODBC est celle lue sans aucune modification. Comme pour le type **char**, une chaîne vide est stockée comme un caractère espace unique, afin de différencier la chaîne vide de la valeur nulle.

champ de type varchar(10)	valeur de data
''	0x32000000000000000000
' '	0x32000000000000000000
'A '	0x41202020202000000000
'AaBb'	0x41614262000000000000
null	0x00000000000000000000

➤ Le type binary

Ce type correspond au type SQL_BINARY dans ODBC. Il permet de stocker de 1 à 32511 caractères.

Dans le cas de CREATE TABLE lors de l'utilisation de ce type de données, un octet supplémentaire est utilisé. Cet octet peut prendre les valeurs 0 ou 1 afin de différencier la valeur nulle de la valeur vide. Si un champ de type **binary** est utilisé dans la définition d'une clef, l'octet notnull_data sera intégré avec ceux de data dans la clef.



Dans le cas de DEFINE TABLE, aucun octet supplémentaire est ajouté. Seul les octets significatifs seront stockés. Ceci interdit donc la différenciation de la valeur 0 de la valeur nulle. C'est pourquoi dans DEFINE TABLE ce type de champ est nécessairement non nullable.

Lors de l'insertion d'une donnée dans ces champs, la zone correspondante du fichier est comblée sur toute sa longueur par des caractères '\0'. Lors de la lecture d'un champ de ce type, tous les octets sont récupérés dans ODBC.

Le tableau suivant donne les valeurs stockées. Les données en italique n'ont de sens que dans le cas de CREATE TABLE.

champ de type binary(10)	nonnull_data	valeur de data
0x	1	0x00000000000000000000
0x00	1	0x00000000000000000000
0x1234	1	0x12340000000000000000
null	0	0x00000000000000000000

➤ Le type varbinary

Ce type correspond au type SQL_VARBINARY dans ODBC. Il permet de stocker de 1 à 32511 caractères.

Ce type ne peut être utilisé que dans la commande CREATE TABLE. Dans DEFINE TABLE, aucun champ de type **varbinary** ne peut être utilisé. Lors de l'utilisation d'un tel type, deux octets supplémentaires sont utilisés. Ces octets contiennent sous la forme d'un entier court, la longueur de la donnée binaire plus un. Ainsi la valeur 0 correspond à un binaire nul, et la valeur 1 à un binaire de longueur 0.

Comme pour le type **binary**, lors de l'insertion d'une donnée dans ces champs, la zone correspondante du fichier est comblée sur toute sa longueur par des caractères '\0'.

Lors de la lecture d'un champ de ce type, seuls les octets sur la longueur du binaire sont récupérés dans ODBC. Si un champ de type **varbinary** est utilisé dans la définition d'une clef, les octets de size_data ne seront pas intégrés avec ceux de data dans la clef.

champ de type varbinary(10)	size_data	valeur de data
0x	0x0001	0x00000000000000000000

0x00	0x0002	0x00000000000000000000
0x1234	0x0003	0x12340000000000000000
null	0x0000	0x00000000000000000000

➤ Le type smallint

Ce type correspond au type SQL_SMALLINT dans ODBC. Il permet de stocker des entiers sur 2 octets. Les valeurs possibles sont de -32767 à 32767. La valeur -32768 est une valeur réservée pour un champ nul.

champ de type smallint	valeur de data
-32767	-32767
0	0
30000	30000
null	-32768

➤ Le type longint

Ce type correspond au type SQL_INTEGER dans ODBC. Il permet de stocker des entiers sur 4 octets. Les valeurs possibles sont de -134217727 à 134217727. La valeur -134217728 est une valeur réservée pour un champ nul.

champ de type longint	valeur de data
-32767	-32767
0	0
134217	134217
null	-134217728

➤ Le type real

Ce type correspond aux types SQL_REAL et SQL_FLOAT dans ODBC. Il permet de stocker des réels flottants sur 4 octets au format machine.

champ de type real	valeur de data
-25	(float)-25.0
0	(float)0.0
3.1415	(float)3.1415
null	valeur non significative



➤ Le type double

Ce type correspond au type SQL_DOUBLE dans ODBC. Il permet de stocker des réels flottants sur 8 octets au format machine.

champ de type double	valeur de data
-25	(double)-25.0
0	(double)0.0
3.1415	(double)3.1415
null	valeur non significative

➤ Le type decimal

Ce type correspond au type SQL_DECIMAL dans ODBC. Il permet de stocker des numériques à virgule fixe.

Le type decimal doit être suivi de deux paramètres entre parenthèses, de la forme "c1 decimal(n, m)" où n représente le nombre total de chiffres et m le nombre de décimales. Si m n'est pas précisé, la valeur par défaut de m est 0.

Les fonctions C-ISAM stdecimal() et lddecimal() sont utilisées pour l'écriture/lecture des champs de ce type.

➤ Le type date

Ce type correspond au type SQL_DATE dans ODBC. Il permet de stocker sur un entier d'une longueur de 4 octets une date correspondant au nombre de jours depuis le 1^{er} janvier de l'an 0. Pour insérer ou tester une date dans les ordres SQL, la notation d'ODBC est à utiliser ({ d 'AAAA-MM-JJ' }).

champ de type date	valeur de data
{ d '0000-01-01' }	0
{ d '1997-02-17' }	729438
null	-134217728

➤ Le type time

Ce type correspond au type SQL_TIME dans ODBC. Il permet de stocker sur un entier d'une longueur de 4 octets une heure correspondant au nombre de seconde dans la journée. Pour insérer ou tester une heure dans les ordres SQL, la notation d'ODBC est à utiliser ({ t 'hh :mm :ss' }).

champ de type time	valeur de data
{ t '00 :00 :00' }	0
{ t '13 :40 :10' }	49210
null	-134217728

➤ Le type timestamp

Ce type correspond au type SQL_TIMESTAMP dans ODBC. Il permet de stocker sur un entier d'une longueur de 4 octets une heure correspondant au nombre de secondes depuis le 1^{er} janvier 1970 (idem la fonction time() en langage C). La valeur maximale correspond à la date du 5 février 2036, 0h. Pour insérer ou tester un timestamp dans les ordres SQL, la notation d'ODBC est à utiliser ({ ts 'AAAA-MM-JJ hh :mm :ss' }).

champ de type timestamp	valeur de data
{ ts '1970-01-01 00 :00 :00' }	0
{ ts '1997-02-17 13 :40 :10' }	856186810
null	-134217728



INDEX

A

Allowed,91
Arrayfields (Progress),96

B

BackOffice,2
Bases de données virtuelles,57
binary,140, 142
bit,139, 140
byte,139, 140

C

Caractères accentués,38
Caractères spéciaux,38
Champ virtuel,60, 68
char,139, 141
C-ISAM,41
 Create database,43
 ISAM-PATH,44
 sqltools,43
 SysColumns,42
 SysDefaults,42
 SysIndexes,42
 SysTables,42
Clauses SQL/C-ISAM
 DEFAULT,111
 FROM,129
 GROUP BY,131
 HAVING,132
 NOT NULL,112
 ORDER BY,133
 SELECT CLAUSE,127
 SET,138
 VALUES,136
 WHERE,130
COLUMN DEFINITION,110
COMMENT,118
CONFIG.XXX,88
CONNECT DATABASE,120
CONSTRAINT DEFINITION,113, 114
Create database,43
CREATE DATABASE,107
CREATE INDEX,116

CREATE SYNONYM,117
CREATE TABLE,108, 139

D

date,140, 145
DB2,17
DBMAP.EXE,93
DBSCRIPT.EXE,94
DBSHOW.EXE,95
Debug,100
decimal,140, 145
DEFAULT,111
DEFINE TABLE,109, 139
DELETE,134
Démonstration,33
Denied,91
DISCONNECT DATABASE,121
double,140, 145
Driver ODBC virtuel,17, 61
DROP DATABASE,119
DROP INDEX,122
DROP SYNONYM,124
DROP TABLE,123, 125

E

Embedded SQL,11
Environnement,60, 66
Exportation d'une source de données virtuelle,79

F

Fichiers
 Fichiers .dat,41, 42, 44, 45
 Fichiers .idx,41, 42, 44, 45
Fichiers C-ISAM,41
Fichiers séquentiels,42
FILE IS,115
FROM,129

G

GROUP BY,131

	H	
HAVING,132		DROP SYNONYM,124 DROP TABLE,123, 125 INSERT,135 UPDATE,137
	I	P
Importation d'une source de données,65		PARAM.XXX,96 Progress,17
Index (C-ISAM),47		
Informix,17, 99		R
INSERT,135		
ISAM-PATH,44		
	J	
Jointure,59		real,140, 144 Redéfinition de base de données,17 Revamping,16, 17, 60
	L	S
Liens inter-tables,73		Script (lancement depuis sqltools),53
longint,140, 144		Sécurité,91, 100
	M	SELECT,126
Modèle client/serveur,13		SELECT CLAUSE,127
	N	SET,138
NIS,19		SGBDR,57
NOT NULL,112		smallint,140, 144
	O	Source de données,26
ODBC,11, 30		Source de données virtuelle,35
Options SQL/C-ISAM		Sous-ensemble SQL/C-ISAM
COLUMN DEFINITION,110		CONSTRAINT DEFINITION,113
CONSTRAINT DEFINITION,114		SQL,14
FILE IS,115		SQL_BINARY,140, 142
Oracle,17, 99		SQL_BIT,139, 140
ORDER BY,133		SQL_CHAR,139, 141
Ordres SQL/C-ISAM		SQL_DATE,140, 145
COMMENT,118		SQL_DECIMAL,140, 145
CONNECT DATABASE,120		SQL_DOUBLE,140, 145
CREATE DATABASE,107		SQL_FLOAT,140, 144
CREATE INDEX,116		SQL_INTEGER,140, 144
CREATE SYNONYM,117		SQL_REAL,144
CREATE TABLE,108		SQL_SMALLINT,140, 144
DEFINE TABLE,109		SQL_TIME,140, 146
DISCONNECT DATABASE,121		SQL_TIMESTAMP,140, 146
DROP DATABASE,119		SQL_TINYINT,139, 140
DROP INDEX,122		SQL_VARBINARY,140, 143
		SQL_VARCHAR,140, 142
		sqltools,43
		Statements
		DELETE,134
		SELECT,126
		Sybase,17, 100
		SysColumns,42
		SysDefaults,42



SysIndexes,42
SysTables,42

T

Table virtuelle,60, 67
Tables (C-ISAM),45
Tables de conversion,38
time,140, 146
timestamp,140, 146
Trace,99
Translateur,31
Tun SQL,16
TUNODBC200.XXX,98
Type
 binary,140, 142
 bit,139, 140
 byte,139, 140
 char,139, 141
 date,140, 145
 decimal,140, 145
 double,140, 145
 longint,140, 144
 real,140, 144

smallint,140, 144
time,140, 146
timestamp,140, 146
varbinary,140, 143
varchar,140, 142

U

UPDATE,137

V

Validation,78
VALUES,136
varbinary,140, 143
varchar,140, 142

W

WHERE,130
WOSA,12