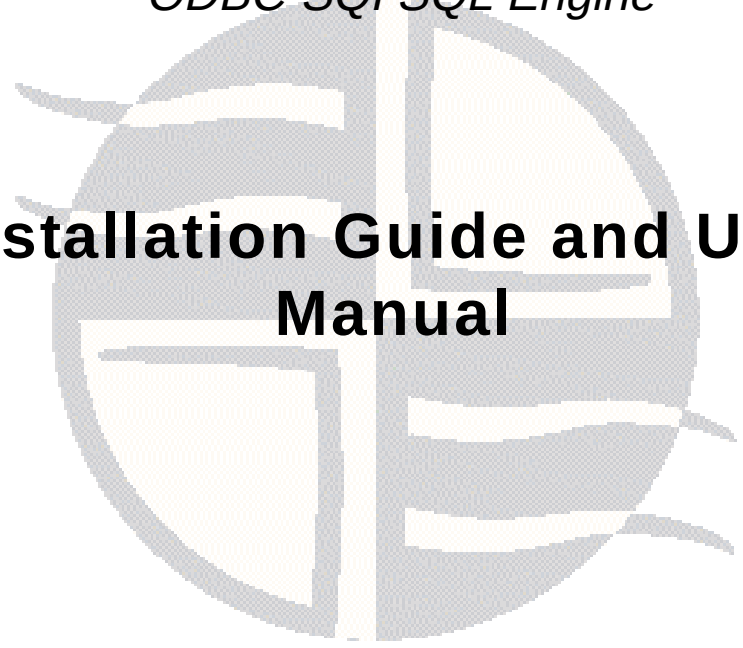
Easysoft® Data Access

ODBC-SQI SQL Engine



Installation Guide and User Manual

Version 14.

Publisher: Easysoft Limited

Thorp Arch Grange

Thorp Arch

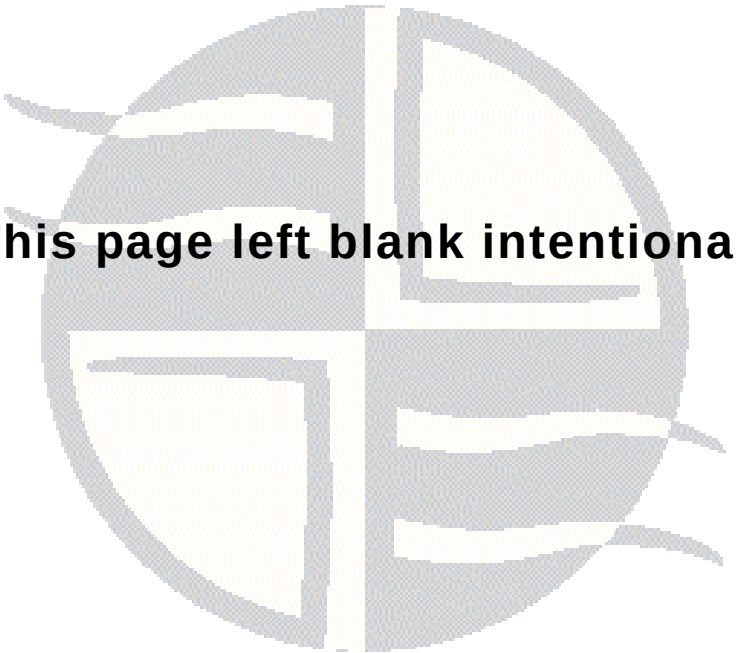
Wetherby

LS23 7BA

United Kingdom

CONTENTS

LIST OF FIGURES



This page left blank intentionally

PREFACE

About this manual

This manual is intended to cover the full range of requirements for anyone wishing to install and use the Easysoft ODBC-SQI SQL Engine.

Chapter Guide

Intended Audience

If you have obtained the Easysoft ODBC-SQI SQL Engine as a component of an Easysoft Data Access product then the information in this manual will be relevant to you, but you should follow the installation and configuration sections of the manual for that specific product.

The manual requires some familiarity with the use of buttons, menus, icons and text boxes. If you have any experience of Apple Macintosh computers, Microsoft Windows or the X Window System, you will have no difficulty with these sections.

NB	In addition to this manual, further information may be provided in README and other files copied into the Easysoft ODBC-SQI SQL Engine installation directory. Please check all the documentation provided before contacting Easysoft with a query.

Displaying the Manual

This manual is available in the following formats:

- Portable Document Format (PDF), which can be displayed and printed using the Acrobat Reader, available free from Adobe at **http://www.adobe.com**.
- HTML (the format Easysoft recommend for viewing onscreen).

Notational Conventions

Across the range of Easysoft manuals you will encounter passages that are emphasized with a box and a label.

A *note box* provides additional information that may further your understanding of a particular procedure or piece of information relating to a particular section of this manual:

NB	Note boxes often highlight information that you may need to be aware of when using a particular feature.

A *reference box* refers to resources external to the manual, such as a useful website or suggested reading:

REF	For more manuals that use this convention, see the rest of the Easysoft documentation.

A *platform note* provides platform-specific information for a particular procedure step:

Linux	In Linux you must log on as the `root` user in order to make many important changes.

A *caution box* is used to provide important information that you should check and understand, prior to starting a particular procedure or reading a particular section of this manual:

Caution!	Be sure to pay attention to these paragraphs because Caution boxes are important!

Typographical Conventions

To avoid ambiguity, typographic effects have been applied to certain types of reference:

- User interface components such as icon names, menu names, buttons and selections are presented in bold, for example:

  Click **Next** to continue.

  Where there is a chain of submenus, the following convention is used:

  Choose **Start > Programs > Command Prompt**.

- Commands to be typed are presented using a `monotype` font, for example:

  At the command prompt type `admin`.

- Keyboard Commands

  It is assumed that all typed commands will be committed by pressing the *<Enter>* key, and as such this will not normally be indicated in this manual. Other key presses are italicized and enclosed by angle brackets, for example:

  Press *<F1>* for help.

- File listings and system names (such as file names, directories and database fields) are presented using the `monotype` plain text style.

Contents

Trademarks

Throughout this manual, *Windows* refers generically to Microsoft Windows 95, 98, 2000, NT or XP, which are trademarks of the Microsoft Corporation. The X Window system is specifically excluded from this and is referred to as *The X Window System* or just *X*.

Easysoft and Easysoft Data Access are trademarks of Easysoft Limited.

1 INTRODUCTION

Introducing the Easysoft ODBC-SQI SQL Engine

The Easysoft ODBC-SQI SQL Engine is a relational database engine that enables heterogeneous access to multiple data sources.

To a client application it is an ODBC 3.5 driver that enhances the functionality of any existing ODBC driver to ODBC 3.5.

The Easysoft ODBC-SQI SQL Engine is a core component of the Easysoft Data Access suite of middleware products.

It can be used with the Easysoft ODBC-ODBC Bridge and the Easysoft JDBC-ODBC Bridge to enable heterogeneous access to multiple local or remote ODBC data sources from non-Windows platforms and Java based applications.

Chapter Guide

The Easysoft ODBC-SQI SQL Engine product component

The Easysoft ODBC-SQI SQL Engine is available both as a separate product and as a component of the following Easysoft Data Access solutions:

- Easysoft Data Access for ISAM
- Easysoft Data Access for Sage Tetra CS/3
- Easysoft Data Access for Zortec System Z
- Easysoft Data Access for Unisys LINC Developer

With these solutions, the Easysoft ODBC-SQI SQL Engine is usually used in conjunction with either the Easysoft ODBC-ODBC Bridge or the Easysoft JDBC-ODBC Bridge to enable cross-platform data access.

This manual explains the installation, configuration and technical aspects of the Easysoft ODBC-SQI SQL Engine when used as a separate product.

If you have obtained the Easysoft ODBC-SQI SQL Engine as a component of an Easysoft Data Access solution, the technical aspects covered in **“Technical Reference” on page 61** are still relevant, but for installation and configuration details you should refer to the manual specific to that Easysoft Data Access solution.

Figure 5 on page 15 shows how the Easysoft ODBC-SQI SQL Engine works in conjunction with other Easysoft Data Access products.

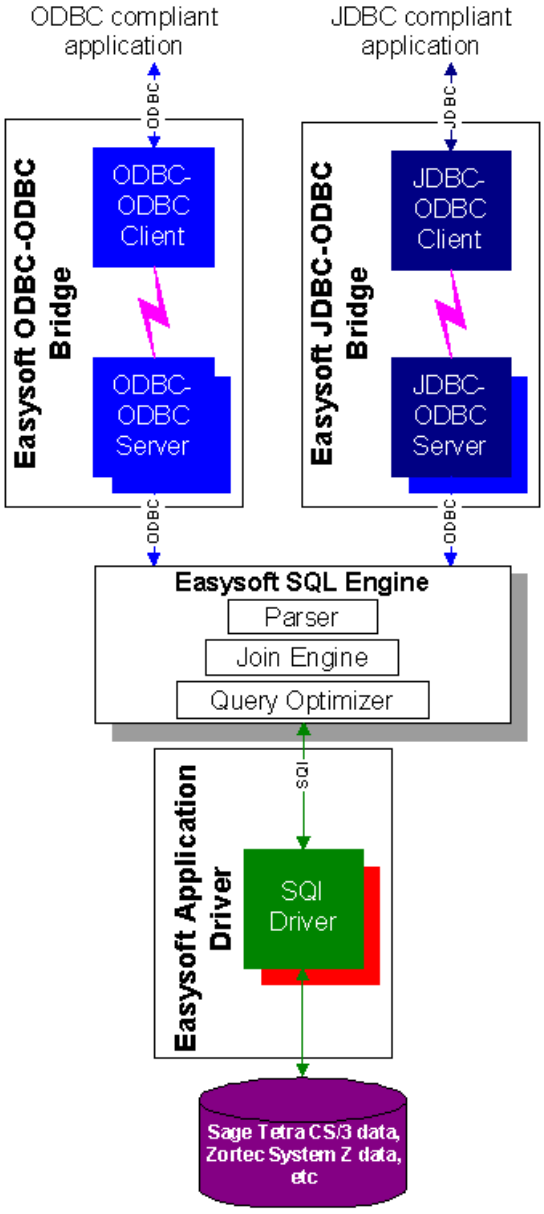


Figure 1: The Easysoft ODBC-SQI SQL Engine product component

Benefits of using the Easysoft ODBC-SQI SQL Engine

An application accesses a data source via an ODBC 'driver'.

For example, suppose that you are updating your employee records. In your ODBC application, you might choose an option to view all employees in the Sales department.

This command is passed to a Driver Manager which loads the appropriate ODBC driver and sends function calls to it.

The ODBC driver then accesses the data and retrieves the appropriate records which are displayed on the application screen:

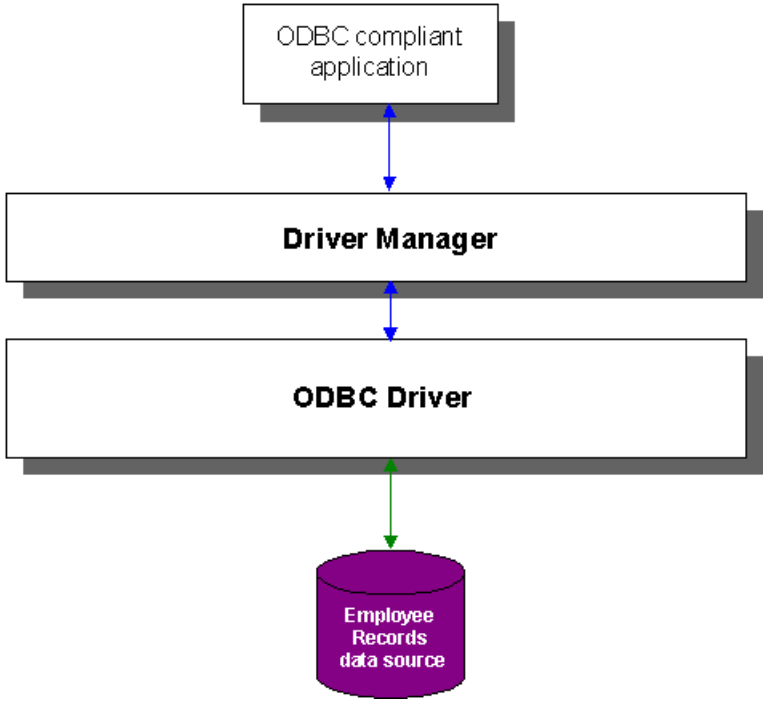


Figure 2: The Easysoft ODBC-SQI SQL Engine architecture

Your work may involve the searching, updating or deleting of records on several different sets of data (or data sources), such as:

- employee records stored in a Microsoft Access database
- product data stored in a Microsoft Excel spreadsheet
- customer information stored in legacy C-ISAM files

Problems that can arise with this scenario include:

- your ODBC application may not be able to query more than one data source at a time
- when performing joins on multiple tables, all the data in those tables has to be brought across the network to your ODBC application, draining network resources
- the drivers that connect to the different data sources may have different functionality or may not support particular SQL constructs, such as outer joins
- your queries may take an unreasonably long time to execute

The following sections explain how the Easysoft ODBC-SQI SQL Engine overcomes these problems.

JOINING MULTIPLE DATA SOURCES INTO ONE

The Easysoft ODBC-SQI SQL Engine enables the seamless joining of data from two or more ODBC data sources.

Applications can query, join and modify data from disparate data sources as though the tables were in a single local data source, known as a *virtual data source*:

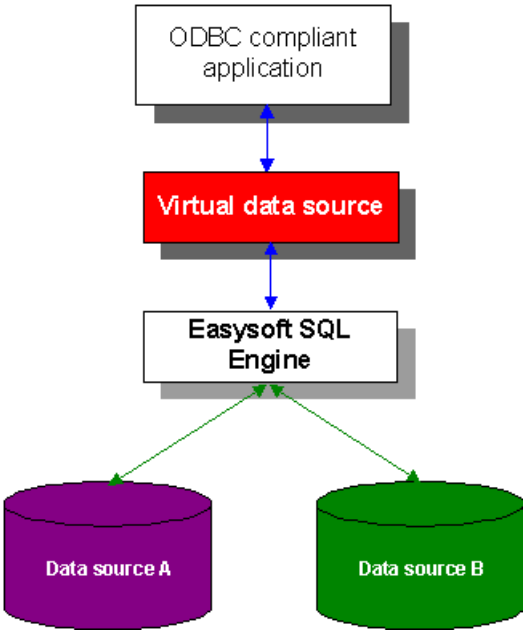


Figure 3: Joining multiple data sources into one

REDUCING NETWORK TRAFFIC

The Easysoft ODBC-SQI SQL Engine performs queries where the data is stored, not where your ODBC application is located.

If your data is stored on a remote machine (the server machine) then SQL queries are performed on that machine, so that only the data in the results set is transferred back across the network to your ODBC application, thus keeping network traffic to a minimum:

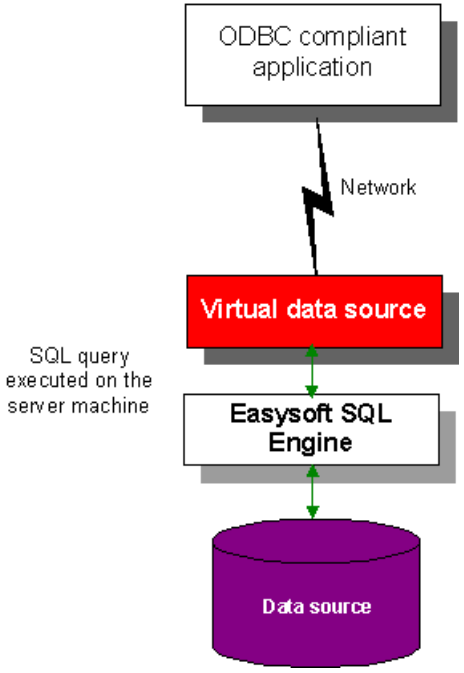


Figure 4: Reducing network traffic

Without the Easysoft ODBC-SQI SQL Engine, queries are performed on the local machine where your ODBC application is running, which can involve transferring large amounts of data across the network, consuming network resources.

ENHANCING THE FUNCTIONALITY OF EXISTING ODBC DRIVERS

The Easysoft ODBC-SQI SQL Engine enhances the functionality of existing ODBC drivers to ODBC 3.5, so that any differences in ODBC conformance between the drivers are irrelevant.

For example, if your existing drivers conform to ODBC 1 or ODBC 2, the Easysoft ODBC-SQI SQL Engine effectively upgrades them to conform to ODBC 3.5:

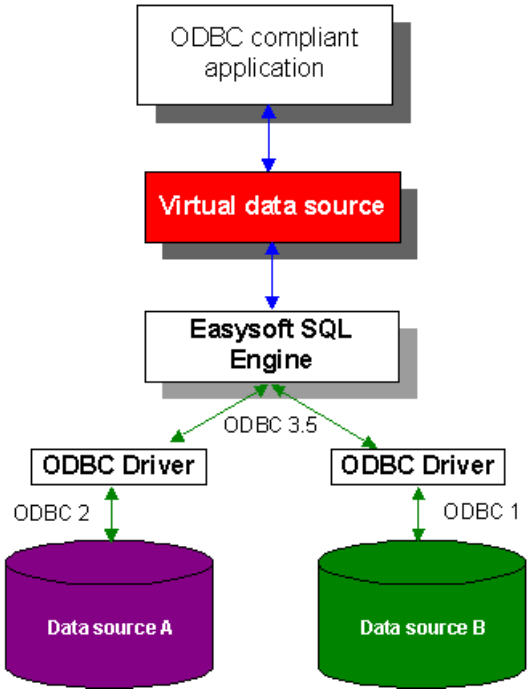


Figure 5: Enhancing the functionality of existing ODBC drivers

QUERY OPTIMIZATION

The Easysoft ODBC-SQI SQL Engine optimizes the syntax of your SQL queries to reduce query times to as short a time as possible.

Understanding how your data is queried

Databases can store vast amounts of data and queries are used to search through the data and find exactly the information that you want to look at or edit.

If you had a database of employee records there would probably many ways in which you would want to edit your employee records.

For example:

- Add and delete employees
- Change an employee's salary
- Change an employee's address
- Record notes following an employee's annual appraisal

Depending on how the database has been implemented, you might be able to perform these edits via a user interface by selecting particular options or typing into certain fields and then clicking a button to commit the changes.

Behind the scenes, the user interface invokes ODBC function calls to connect to the data source and execute SQL statements to perform the edit.

ODBC is a *programming interface* that enables applications to connect to relational databases. SQL is a *data sublanguage* used for manipulating data in relational databases. *SQL statements are embedded as arguments of ODBC function calls* to perform operations on data.

You can write SQL queries manually, and some advanced users might prefer to do this rather than manipulate data via a user interface.

In some instances, it might not be possible to perform a complex query through the user interface available so the only alternative is to write the SQL query by hand.

A simple SQL query might be:

```
SELECT * FROM EMPLOYEE WHERE DEPT='SALES'
```

which would display the details of all employees in the Sales department.

There are different levels of ODBC and SQL conformance.

The Easysoft ODBC-SQI SQL Engine conforms to ODBC 3.5 and supports the SQL minimum grammar (which is a recognized standard) with the majority of SQL-92 extensions.

Conformance within the Easysoft ODBC-SQI SQL Engine is documented fully in **“Technical Reference” on page 61**, where you will also find details of standards and specifications for ODBC and SQL.

If you are a developer implementing bespoke ODBC applications or customizations for manipulating relational databases, refer to the **“Technical Reference” on page 61** for full Easysoft ODBC-SQI SQL Engine technical details.

Reproducing SQL statements

In this documentation there are examples of SQL statements and the results you can expect from particular statements.

When reproducing these SQL statements on your own system, please note that:

- Your SQL tool might require you to end each SQL statement with a semi-colon ';'.

  The semi-colon is a separator which allows you to separate one SQL statement from another in the same command, but some SQL tools expect you to include the semi-colon at the end of all statements.

  For example:

  ```
  select * from table1 ;
  ```

  This trailing semi-colon is not included in the syntax in this documentation, as it is not required by the `isql` tool used as reference.

- For clarity, some sample SQL statements in this manual are set out on multiple lines.

  Depending on the SQL tool that you are using, you might be able to enter statements over multiple lines or you might need to type them on one continuous line.

  Your SQL tool will probably report an error if it cannot process statements extending over multiple lines.

- If you work with table names that contain spaces, you must place quotation marks around the table names in any SQL statements.

For example:

```
select "New Customers".CustID, "Existing
Customers".CustID from "New Customers" and
"Existing Customers"
```

will select the `CustID` data from both the "`New Customers`" table and the "`Existing Customers`" table.

Overview of installation and setup

The main stages in installing and setting up the Easysoft ODBC-SQI SQL Engine are:

1. Download the software from the Easysoft web site (**www.easysoft.com**).
2. Install the Easysoft ODBC-SQI SQL Engine on the machine from where you intend to access the data sources.
3. Set up your virtual data sources.
4. Once you have created your virtual data sources, you can connect to them via any ODBC-compliant application.

   When installing the Easysoft ODBC-SQI SQL Engine for use with the Easysoft ODBC-ODBC Bridge or Easysoft JDBC-ODBC Bridge, there is no particular sequence in which the products should be installed, but you should configure the data sources for accessing your remote data and ensure that they are connecting to the data correctly before including them in a virtual data source.

INSTALLATION

2

Installing the Easysoft ODBC-SQI SQL Engine

This section explains how to install, license and remove the Easysoft ODBC-SQI SQL Engine on supported Windows platforms.

The installation can be carried out by anyone with local administrator privileges for the target machine.

Chapter Guide

Obtaining the Easysoft ODBC-SQI SQL Engine

There are three ways to obtain the Easysoft ODBC-SQI SQL Engine:

- The Easysoft web site is available 24 hours a day at **http://www.easysoft.com** for downloads of definitive releases and documentation.

  Select **Download** from the Easysoft ODBC-SQI SQL Engine section of the website and then choose the platform release that you require.

  First time visitors must complete the new user form and click **Register**. Note that your personal Internet options may require you to login and click **Continue** if you have previously registered.

- The Easysoft FTP server is available 24 hours a day at **ftp://ftp.easysoft.com**, containing free patches, upgrades, documentation and beta releases of Easysoft products, as well as definitive releases.

  The FTP site is useful if you have a slow connection or if you want to write a script to retrieve the file.

  Change to the `pub/sql_engine` directory and then choose the platform release that you require.

- If you have an extremely slow connection you can order Easysoft software on CD by email, telephone or post (see **Contact Details**).

What to install

The selection of components that you require to download in order to configure the Easysoft ODBC-SQI SQL Engine varies depending on the platforms on which you wish to run.

All installations must download the Easysoft ODBC-SQI SQL Engine software itself.

The name of the Easysoft ODBC-SQI SQL Engine distribution file is of the form:

- `EasysoftODBC-SQISQLEngine.exe` (Windows)

Copy the downloaded distribution file into a temporary directory on the machine where you intend to create your virtual data sources.

CROSS-PLATFORM DATA ACCESS

If you intend to implement cross-platform data access, you also need to install either the Easysoft ODBC-ODBC Bridge (see **http://www.easysoft.com/products/2002/main.phtml**) for remote ODBC access or the Easysoft JDBC-ODBC Bridge (see **http://www.easysoft.com/products/2003/main.phtml**) for remote JDBC access from Java applications.

The Easysoft ODBC-ODBC Bridge consists of two separate client and server components and the Easysoft JDBC-ODBC Bridge consists of a single server component.

NB	You cannot 'mix and match' server and client components of the Easysoft ODBC-ODBC Bridge and Easysoft JDBC-ODBC Bridge.

The following components are required for remote ODBC access to multiple databases on different platforms through a virtual data source:

- the Easysoft ODBC-SQI SQL Engine on the client platform
- the Easysoft ODBC-ODBC Bridge client component on the client platform
- the Easysoft ODBC-ODBC Bridge server component *on each database platform*

NB	The first two digits of the version number must match when the Easysoft ODBC-ODBC Bridge client and server components are installed (see the Easysoft ODBC-ODBC Bridge manual for more information). This does not apply to the Easysoft JDBC-ODBC Bridge, where no specific client installation is required.

The following components are required for remote JDBC access to multiple databases on different platforms through a virtual data source:

- the Easysoft ODBC-SQI SQL Engine on the client platform
- the Easysoft JDBC-ODBC Bridge *on each database platform*

Win 9x	If you are using Windows 9x with either the Easysoft ODBC-ODBC Bridge or the Easysoft JDBC-ODBC Bridge you will need Winsock2, which can be downloaded from **http://www.microsoft.com/windows95/downloads/contents/wuadmintools/s_wunetworkingtools/w95sockets2/**.

The required cross-platform data access software can be obtained as follows:

1. From the Easysoft Web site:

For the Easysoft ODBC-ODBC Bridge:

- Download the Easysoft ODBC-ODBC Bridge from **http://www.easysoft.com/products/9999/download_cs.phtml?product=2002**, selecting your required operating system for both client and server.
- Install the **Client Download** onto your client machine.

  This provides you with the Easysoft ODBC-ODBC Bridge Client.
- Install the **Server Download** onto your server machine.

  This provides you with the Easysoft ODBC-ODBC Bridge Server.

For the Easysoft JDBC-ODBC Bridge:

- Download the Easysoft JDBC-ODBC Bridge from **http://www.easysoft.com/products/9999/download.phtml?product=2003**, selecting your required operating system, and install it on your server machine.

  This provides you with the Easysoft JDBC-ODBC Bridge Server.

2. From the Easysoft FTP site:
   - for the Easysoft ODBC-ODBC Bridge both client and server components are contained in the same Easysoft ODBC-ODBC Bridge executable file held in the **ftp://ftp.easysoft.com/pub/odbc-odbc-bridge/** directory.
   - the Easysoft JDBC-ODBC Bridge server component is contained in the Easysoft JDBC-ODBC Bridge executable file held in the **ftp://ftp.easysoft.com/pub/jdbc-odbc-bridge/** directory. There is no client component to install.

NB	Both client and server components are held in the same executable installation file for the Easysoft ODBC-ODBC Bridge. There is no specific client installation for the Easysoft JDBC-ODBC Bridge.

Within your licensed major version number, you should go for the highest release available for your platform.

To install software of a different major version number requires a new license.

You should now download the file and begin the installation process:

- **"Installing the Easysoft ODBC-SQI SQL Engine" on page 31**
- **"Uninstalling the Easysoft ODBC-SQI SQL Engine" on page 42**

Installing the Easysoft ODBC-SQI SQL Engine

For Windows, the Easysoft ODBC-SQI SQL Engine installation comes as an executable named `EasysoftODBC-SQISQLEngine.exe`.

If there is a choice of files then you should download the file with the highest version number.

1. From the web, click to download the distribution file.

   – OR –

   In your FTP client, switch to `binary` mode and `get` the distribution file.

   – OR –

   If you have a CD, navigate to the folder containing the distribution file.

Caution! Please shut down other Windows programs before installing. In particular, Microsoft Outlook can cause the installation routine to pause for several minutes when you start it.

Beginning the Installation

2. Execute the distribution file that you downloaded in **"Obtaining the Easysoft ODBC-SQI SQL Engine" on page 26**.

There will be a short delay while setup prepares the wizard to guide you through the rest of the install procedure before the **Welcome** dialog box is displayed:

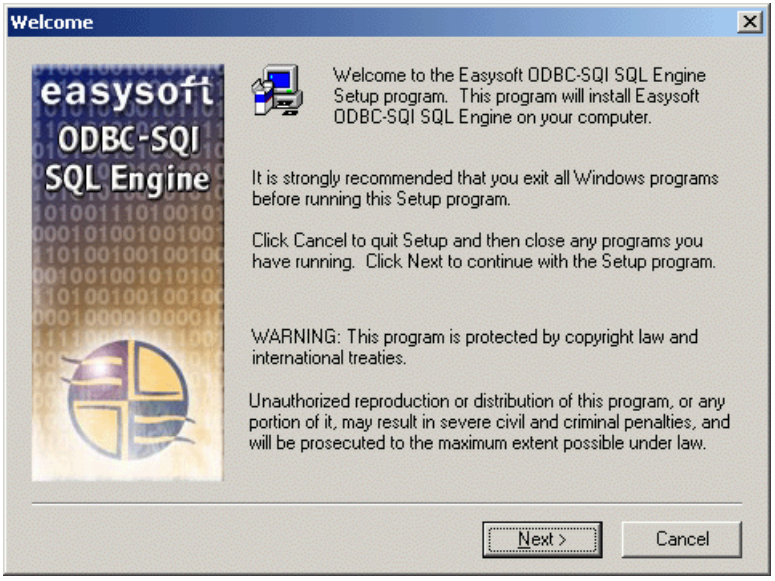


Figure 6: The Welcome dialog box

3. Click **Next** to continue.

The **Software License Agreement** dialog box then displays Easysoft End User licensing details:

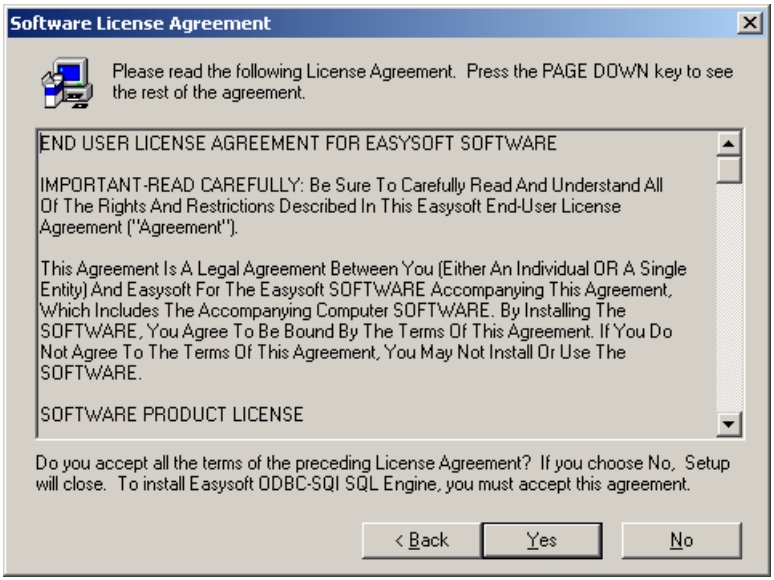


Figure 7: The Software License Agreement dialog box

You are required to accept the terms of the License Agreement before continuing.

4. If you do not agree to the License Agreement, click **No** to exit the installation.

   – OR –

   Click **Yes** to accept the License Agreement and continue with the installation.

The **Information** dialog box is displayed:

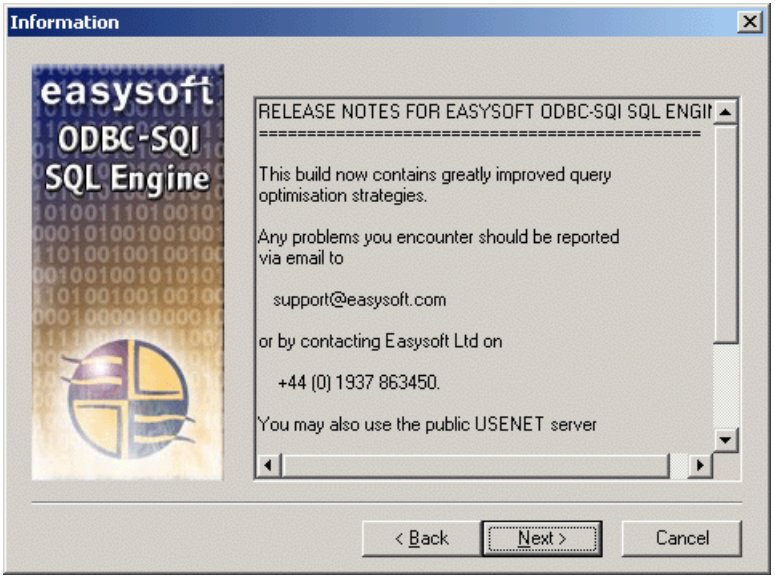


Figure 8: The Information dialog box

5. Click **Next** to continue.

The **Choose Destination Location** dialog box is displayed:

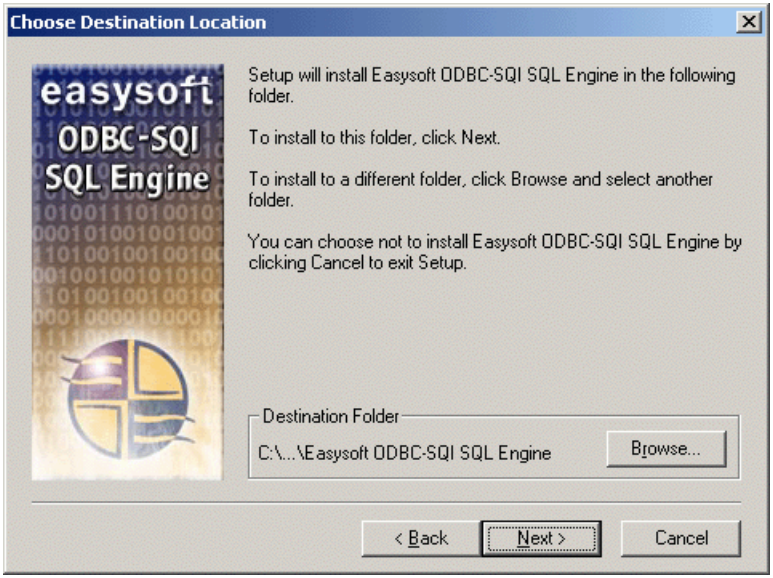


Figure 9: The Choose Destination Location dialog box

Choose the directory in which you want to install the Easysoft ODBC-SQI SQL Engine.

6. To accept the default, click **Next**.

   – OR –

   To choose an alternative directory, click **Browse** to select the path you want, then click **Next** to continue.

   There is now a short wait while the relevant Easysoft ODBC-SQI SQL Engine components are copied and configured.

   The install program now starts the Easysoft License Manager (explained fully in the **Licensing Guide**).

The following types of license are available:

- a *free time-limited trial license* which gives you unrestricted use of the product for a limited period (usually 30 days).
- a *full license* if you have purchased the product. On purchasing the product you are given an authorization code which you should have to hand.

By licensing the Easysoft ODBC-SQI SQL Engine you are given three licenses: one for the Engine and two for SQI drivers which are components of the Engine.

Simple Query Interface (SQI) is an Easysoft term referring to the interface closest to the actual data being accessed. It allows the Easysoft ODBC-SQI SQL Engine to access that data.

For example, the Easysoft SQI-Tetra driver contains an SQI that enables the Easysoft ODBC-SQI SQL Engine to access data stored in Sage Tetra CS/3.

7. Enter your contact details.

   You MUST enter the **Name**, **E-Mail Address** and **Company** fields.

   The **Telephone** and **Facsimile** fields are important if you require Easysoft to contact you by those methods.

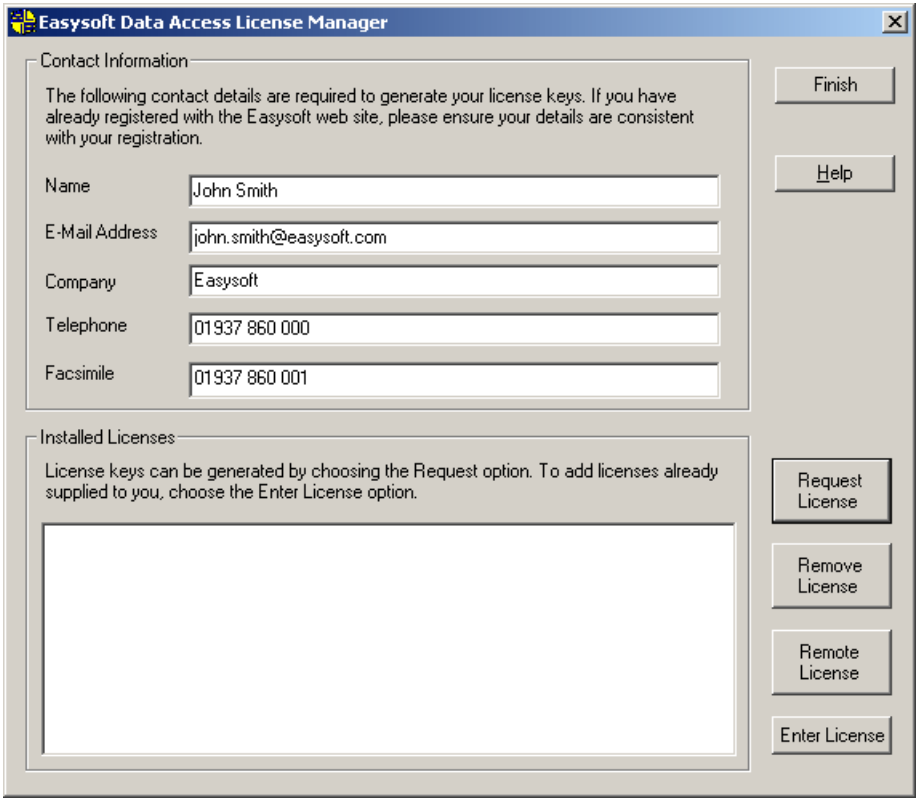


Figure 10: The Windows License manager data entry screen

8. Click **Request License**.

   You are asked for a license type.

9. To trial the product, choose **Time Limited Trial** and then click **Next**.

   The License Manager asks what software you are licensing. Select `Easysoft SQL Engine` from the drop-down list and then click **Next**.

   – OR –

If you have purchased the software and received your authorization code, select **Non-expiring License** and then click **Next**. The License Manager requests your authorization code. Enter the authorization code and then click **Next**.

10. The License Manager displays a summary of the information you have entered and lets you choose how to apply for the license.

Figure 11: The Windows License manager summary screen

Choose **On-line Request** if your machine has a connection to the internet. The License Manager then transmits a network packet to the license server at Easysoft. The whole process is automatic and invisible, and you can proceed to **step 11 on page 39**.

NB	Only your license request identifier and contact details as they appear in the License Manager main screen are sent to Easysoft.

The remaining three options (**Email Request**, **Print Request** and **View Request**) are all ways to obtain a license if your machine is offline (i.e. does not have a connection to the internet). Each of these methods involves providing Easysoft with information including your site number (a number unique to your machine), then waiting to receive your license key. Instead of emailing, faxing or telehoning your details to Easysoft, you can enter them directly onto Easysoft's web site and your license key will be emailed to you automatically. To use this method, click **View Request** to display your site number, then run a web browser and go to **http://www.easysoft.com/sales/autolicense.phtml**. Choose the type of license you require and enter your site number and then click **Continue**. Your license key will now be emailed to you. When you receive the license key, you can activate it either by double-clicking the email attachment or by clicking **Enter License** on the License Manager main screen and pasting the license key into the dialog box. A message is displayed, telling you how many licenses have been added.

NB	If you use the **Email Request** option, the license key is emailed to the email address as displayed on the License Manager main screen, not the `from:` address of your email.

11. Click **Finish** in the License Manager to return to the install program.

The **Setup Complete** dialog box is displayed:

Figure 12: The Setup Complete dialog box

12. Click **Finish**.

    The installation is complete.

    A folder called **Programs > Easysoft > ODBC-SQI SQL Engine** is created within the Windows Start menu containing links to:

    - the Easysoft ODBC-SQI SQL Engine user guide in Help format
    - the Easysoft ODBC-SQI SQL Engine news group
    - the Easysoft web site

TESTING THE INSTALLATION

You can check whether the installation has worked by linking to data in a sample virtual data source which is created during the installation. The sample data is installed into the `\Demo` subdirectory under the install path.

The following steps explain how to link to the sample virtual data source from Microsoft Access, but you could use any ODBC application on your machine (this example uses Microsoft Access 2000; you may need to adapt these instructions for other versions).

1. Run Microsoft Access.
2. Create a new blank database.
3. Display the **Tables** tab on the database window.
4. Right-click in the empty window, then select **Link Tables**. The **Link** dialog box is displayed.
5. From the **Files of Type** drop-down list box, select **ODBC Databases ()**. The **Select Data Source** dialog box is displayed.
6. Click the **Machine Data Source** tab.
7. Select the `SQL_Engine` data source.
8. Click **OK**. The **Link Tables** dialog box is displayed.
9. Click **Select All** to link all the tables into the database, then click **OK**. After a few moments, the tables are listed in the **Tables** tab of the database window in Microsoft Access.
10. Double-click a table to view its data.

    Connecting to your own virtual data sources is explained in **"Configuration and Connection" on page 45**.

Uninstalling the Easysoft ODBC-SQI SQL Engine

This section explains how to remove the Easysoft ODBC-SQI SQL Engine from your system although you do not normally need to uninstall this software before installing a more recent version.

1. Select **Start > Settings > Control Panel**, then double-click the **Add/Remove Programs** icon.

   You are then presented with a list of applications that can be removed automatically.

2. Select **Easysoft SQL Engine** and then click **Add/Remove**.

3. Click **Yes** to confirm that you wish to remove the Easysoft ODBC-SQI SQL Engine and all its components.

   The system begins to remove all the components. If shared components seem not to be required, you will be prompted to decide whether or not to delete them.

NB	Windows' install/uninstall procedures incorporate a mechanism in the registry to determine whether or not shared files are still required by other programs. Sometimes this database can become out of date, for example if the user deleted an application directly, without using **Add/Remove Programs**, or the registry was 'repaired' after a system crash.

4. If you feel confident with the registry (i.e. your system has had relatively few programs installed and removed) you should click **Yes** or **Yes to all** to continue.

   – OR –

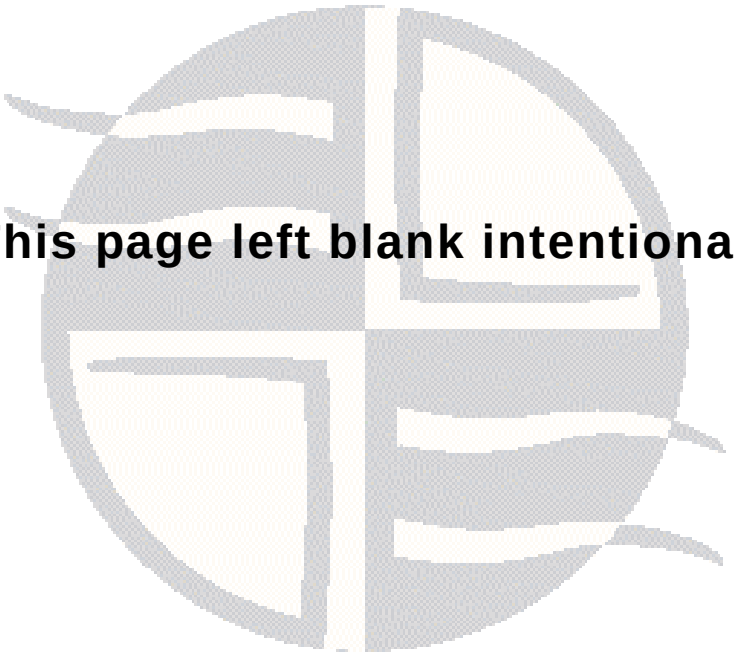
   If you have any doubts (e.g. uninstall procedures have failed in the past) you should click **No** or **No to All**.

The uninstall process removes the Easysoft ODBC-SQI SQL Engine components from your system.

NB	If files have been created in any of the installation directories then these directories will not be removed. In this case, the uninstall program will issue a warning and you can click **Details** to find out which directories remain.

5. On completion, click **OK** to go back to the Control Panel Install/Uninstall window.
6. The uninstall process is complete.

   Note that when you uninstall, your licenses are not removed so you do not need to relicense the product if you reinstall or upgrade.



This page left blank intentionally

CONFIGURATION AND CONNECTION

3

Configuring and connecting to virtual data sources

This section explains how to configure a virtual data source using the Easysoft ODBC-SQI SQL Engine and how to connect to a virtual data source from an ODBC-compliant application.

An exercise is provided which takes you step-by-step through configuring a virtual data source which joins local and remote data sources and then executing SQL queries on this virtual data source.

Chapter Guide

User data sources and system data sources

You can create two types of data sources: user data sources and system data sources:

- a user data source is only visible to the user who was logged into the machine when the data source was created
- a system data source is visible to any user, or service, logged into the machine.

When creating virtual data sources, you could create a virtual system data source that includes some user data sources. If you do this, be aware that connecting to the virtual data source will fail if you are not logged in as the appropriate user.

Connecting to remote data sources

If you are using the Easysoft ODBC-ODBC Bridge or the Easysoft JDBC-ODBC Bridge to implement cross-platform data access, you should first use those products to configure data sources connecting to the remote data.

Once those data sources are configured *and proven to be connecting to the remote data*, you can include them in a virtual data source.

Please refer to the documentation provided with the Easysoft ODBC-ODBC Bridge or the Easysoft JDBC-ODBC Bridge for details of connecting to remote data.

Connecting to multiple data sources

The Easysoft ODBC-SQI SQL Engine provides a mechanism to avoid any conflicts that might arise if you combine two or more data sources into one virtual data source and the same table name is used in more than one of the back-end data sources.

For example, imagine that a company has two offices (one for the northern region and one for the southern region) and both offices use the same database structure:

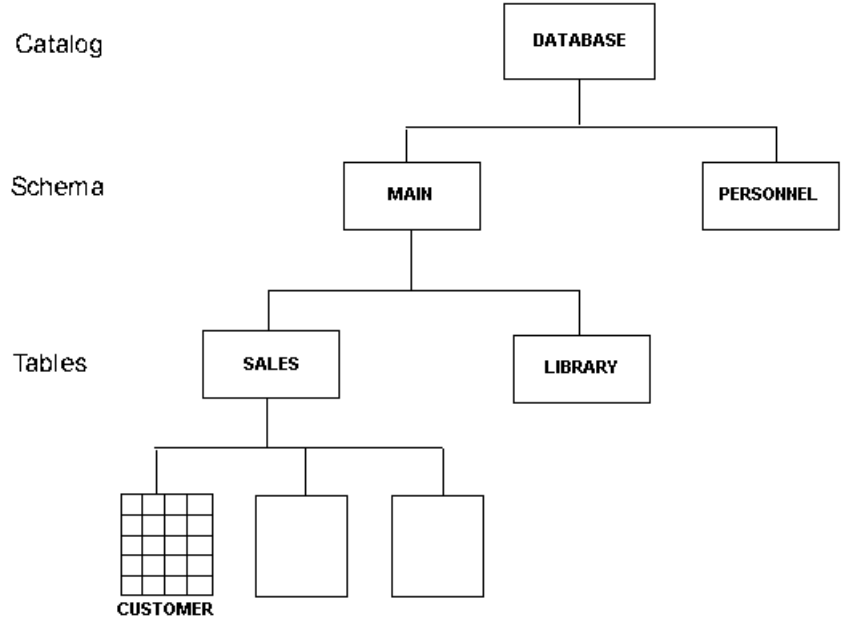


Figure 13: Connecting to multiple data sources

Each database is split into two catalogs: MAIN and PERSONNEL.

The MAIN catalog contains two schemas: SALES storing all sales information, and LIBRARY containing details of all books and

software held at that office. The SALES schema contains multiple tables including the CUSTOMER table which contains details of all the customers at that office.

Catalogs and *schemas* are levels in the containment hierarchy of a database's structure. A database can contain one or more catalogs; each catalog can contain one or more schemas; each schema can contain multiple tables.

When the two offices merge together, their individual databases are combined into one virtual data source but there needs to be some way of distinguishing between, for example, the customers at each office. With the Easysoft ODBC-SQI SQL Engine you can do this using *link names*.

A link name is prefixed to the catalog name of the data source, ensuring uniqueness for each back-end data source. You can then reference a specific table using the following name syntax:

link_name.catalog_name.schema_name.table_name

For example, if the link names NORTH_OFFICE and SOUTH_OFFICE were created, you would reference the CUSTOMER table in the northern office's database using:

```
NORTH_OFFICE.MAIN.SALES.CUSTOMER
```

You might not need to include the catalog and schema names if the back-end DBMS does not require them. If this is the case, you can use the following syntax to reference a specific table:

```
link_name...table_name
```

For example

```
NORTH_OFFICE...CUSTOMER
```

So if you wanted to select the records of all customers in the northern office's database, the SQL statement would be:

```
SELECT * FROM NORTH_OFFICE...CUSTOMER
```

If you know that a table is unique within the virtual data source, there is no need to include the link name when referencing that particular table.

The setting up of link names is explained in the appropriate platform-specific section of this chapter.

Virtual data sources

This section provides instructions for configuring and connecting to virtual data sources:

- **“Configuring a virtual data source” on page 50**
- **“Connecting to a virtual data source” on page 54**
- **“A Sample Exercise” on page 55**

CONFIGURING A VIRTUAL DATA SOURCE

The Easysoft ODBC-SQI SQL Engine can connect to any local ODBC data sources set up on your machine or any remote data sources that you are connecting to via the Easysoft ODBC-ODBC Bridge or the Easysoft JDBC-ODBC Bridge.

In Windows, your data sources are managed using the ODBC Data Source Administrator.

To configure a virtual data source:

1. Select **Start > Settings > Control Panel** and open the ODBC Data Sources icon.

2000	To find the ODBC icon in Windows 2000, open `Administrative Tools` in Control Panel. The ODBC icon is called `Datasources (ODBC)`.

The ODBC Data Source Administrator opens.

2. Click the **User DSN** tab to create a data source which is only available to the user who is currently logged into this Windows machine.

– OR –

Click the **System DSN** tab to create a data source which is available to any user or service that logs into this Windows machine.

3. Click **Add** to create a data source.

   The **Create New Data Source** dialog box lists the drivers available.

4. Select `Easysoft SQL Engine` and then click **Finish**.

   The **Easysoft DSN Setup** dialog box is displays two tab folders:

   - use the **ODBC Gateway** tab to add ODBC data sources to this virtual data source.
   - use the **Configuration** tab to specify paths and memory settings for this virtual data source.

5. On the **ODBC Gateway** tab, enter a name and description for this virtual data source.

   To add an ODBC data source to this virtual data source, click the data source in the **Available DSNs** list so that it is highlighted (or selected), and then click **->** to move it into the **Used DSNs** list.

   To deselect a data source in either the **Available DSNs** or **Used DSNs** list, click it again so that it is not highlighted.

   To remove an ODBC data source from this virtual data source, select it in the **Used DSNs** list then click **<-**.

   Once a data source is listed in **Used DSNs**, you can override any authentication settings on the data source, and change additional settings. To do this, double-click the data source in the **Used DSNs** list. The **DSN Properties** dialog box is displayed.

You can change:

- the data source being connected to (**Target DSN**)
- the user name (**Target UID**) and password (**Target Password**) with which the connection is being made. Any values that you specify here will override any settings on the back-end data source itself. For example, the back-end data source might be configured to allow the '`admin`' user to log on with full edit access, but you could enforce read-only access via the virtual data source by specifying a user (and password) who has read-only privileges to the data.

NB If the back-end DBMS requires authentication, you should always include valid Target UID and Target PWD values, as otherwise some SQL applications will pass the empty Target UID and Target PWD values to the back-end DBMS, ignoring any authentication that may have been specified in the back-end data source. This will result in a failure to connect to the virtual data source.

- additional options to include in the connection, e.g. you could specify a particular database within the data source (**Extra Options**).
- in the **Link Name** box, you can specify a unique name for this data source (see **"Connecting to multiple data sources" on page 47** for more information).
- select the **Disable SQL Passthrough** option to prevent the back-end DBMS from attempting to perform queries and have the Easysoft ODBC-SQI SQL Engine seamlessly take over (e.g. if you know from experience that the database hangs when attempting some queries).

6. On the **Configuration** tab, enter a name and description for this virtual data source if you have not already specified these details on either of the other tabs.

   This tab stores paths of temporary directories and some memory settings. Normally the default settings will be acceptable, but you can change them if necessary.

   - **Sort Working Path** is a temporary directory used during sorts
   - **RS Working Path** is a temporary directory used by results sets
   - **Memo Working Path** is a temporary directory used for storing binary object data
   - **Sort RAM Buffer** is the amount of RAM allocated for sorting
   - **Result Set Cache** is the number of rows that can be cached in RAM.

7. Click **OK** to create this virtual data source.

   You can now connect to this data source via ODBC-compliant applications such as Microsoft Access.

NB	Because of limitations within Microsoft Access, if your virtual data source contains any Microsoft Access data sources, you will not be able to connect to the virtual data source from Access.

CONNECTING TO A VIRTUAL DATA SOURCE

The following steps explain how to connect to a virtual data source from Easysoft iSQL, but you could use any ODBC-compliant application on your machine.

NB	iSQL is currently available from Easysoft as beta software. If you would like a copy of this tool for connecting to and querying ODBC data sources, please contact **sales@easysoft.com**.

If you choose to use an ODBC-compliant application other than Easysoft iSQL, refer to the documentation supplied with that application for details of connecting to a data source.

1. Select **Start > Program Files > Easysoft > Easysoft iSQL**.

   The Easysoft iSQL program window opens.

2. Click the **+** alongside Drivers to expand the list of drivers available.

3. Click the **+** alongside Easysoft ODBC-SQI SQL Engine to view the data sources configured using the Engine.

   Data sources can be User or System data sources. Click the **+** alongside the **User** or **System** label to view the data sources of that type.

4. When you can see the data source that you want to connect to, double-click on it. The status line shows `'connecting to data source'`.

5. Once the connection has been made, to view a list of the tables in this data source, click **Show Tables**.

6. When the **SQL Tables** dialog box appears, click **OK** without changing any of its settings.

   The tables are listed in the lower half of the right-hand pane.

7. Type your SQL statement into the upper half of the right-hand pane. For example, to view the contents of a table, type:

   SELECT * FROM *tablename*

   where *tablename* is the name of a table in the data source.

8. To execute the SQL statement, click **Execute Query**.

   The results are displayed in the lower half of the right-hand pane.

   For more examples of writing SQL statements, work through the **"A Sample Exercise" on page 55**.

A Sample Exercise

This exercise takes you through configuring a virtual data source which joins a local data source and a remote data source, then executing SQL queries on the virtual data source to query both back-end data sources as though they were one.

The local data source

When you install the Easysoft ODBC-SQI SQL Engine, a sample data source called `SQL_Engine` is created, which connects to an Easysoft database on the local machine.

The remote data source

Because there is no way of knowing what remote data sources you might have, this exercise will connect to a remote data source on a server at Easysoft.

To do this, you must install the Easysoft ODBC-ODBC Bridge client component (which is free) on your local machine, and accept the option to create the DEMO data source at the end of the installation.

You can download the Easysoft ODBC-ODBC Bridge from **http://www.easysoft.com/products/oob/download.phtml** (you will be prompted to log into the web site if you are not currently logged in).

Once the Easysoft ODBC-ODBC Bridge client is installed and the DEMO data source is set up, check that you can connect to the DEMO data source before proceeding with the following exercise.

If you cannot connect to the DEMO data source, you will not be able to complete this exercise.

The Easysoft ODBC-ODBC Bridge manual is available in PDFor HTML format at **http://www.easysoft.com/products/9999/documentation.phtml?product=2002** and contains information about installing and setting up data sources for the Easysoft ODBC-ODBC Bridge.

Setting up the virtual data source

Before proceeding, ensure that the SQL_Engine data source and the DEMO data source are correctly set up on the local machine.

To create the virtual data source:

1. Open the **ODBC Data Source Administrator** (see **“Configuring a virtual data source” on page 50** for further information).
2. On the **System DSN** tab, click **Add**.
3. Select the Easysoft ODBC-SQI SQL Engine driver and then click **Finish**.

   The **Easysoft DSN Setup** dialog box is displayed.
4. On the **ODBC Gateway** tab, type `Virtual DSN Test` in the **DSN** box.

5. Select the DEMO data source in the **Available DSNs** list and then click **>** to move it into the **Used DSNs** list.
6. Double-click on DEMO in the **Used DSNs** list.
7. Specify values for the following fields in the **DSN Properties** dialog box:

Data Item	Value
Target DSN	demo
Target UID	demo
Target Password	easysoft

Figure 14: Virtual data source values under Windows

8. Click **OK**.
9. On the **Easysoft RDBMS** tab, select the **Enable Easysoft DB** option and set both the **Schema Path** and **Data Path** to point to the location of the sample files which is `\Program Files\Easysoft\Easysoft SQL Engine\Demo` by default.
10. Click **OK** on the **Easysoft DSN Setup** dialog box and then click **OK** to close the **ODBC Data Source Administrator**.

Querying the virtual data source

In this part of the exercise the Easysoft iSQL utility will be used to query the virtual data source that you have set up.

1. Select **Start > Program Files > Easysoft > Easysoft iSQL**.
2. Click the **+** symbol alongside Drivers - Easysoft SQL Engine - System.

You should be able to see the Virtual DSN Test data source and any other system data sources that you have configured using the Easysoft ODBC-SQI SQL Engine.

3. Double-click on the Virtual DSN Test data source. The status line shows `'connecting to data source'`.

4. Once the connection has been made, click **Show Tables** to display a list of the tables in this data source.

5. Click **OK** on the **SQL Tables** dialog box without changing any of its settings.

   The tables are listed in the lower half of the right-hand pane.

   The TABLE_CAT and TABLE_SCHEM columns identify the 'back-end' data source to which a table belongs:

   - When these columns show `pubs` and `dbo` respectively, they identify a table belonging to the DEMO data source.
   - When TABLE_CAT is empty and TABLE_SCHEM shows `SAMPLE`, they identify a table belonging to the SQL_Engine data source.

   The DEMO data source contains a database of book sales and the SQL_Engine data source contains a database of product sales. The two data sources are unrelated.

   Normally your data sources will have something in common which will be why you want to query them heterogeneously.

6. Type the following SQL statement into the upper half of the right-hand pane.

   ```
   SELECT * FROM SALES
   ```

7. Click **Execute Query** in the iSQL toolbar to execute this statement.

A list of book sales in the DEMO data source is displayed. Notice that the title_id is shown but book titles are not shown because that information is not stored in the SALES table.

8. To include book titles in the results, type:

   ```
   select sales.*, titles.title from sales, titles
   where sales.title_id=titles.title_id
   ```

   Remember to click **Execute Query**.

   This SQL statement is saying 'display all columns in the SALES table, and the TITLE column from the TITLES table, where the TITLE_ID number is the same in both the SALES and TITLES tables'.

   The same results are returned, but this time book titles are included.

9. To view a list of orders in the SQL_Engine data source, type:

   ```
   SELECT * FROM SAMPLE.orders
   ```

   1500 records are returned.

10. To make this query more specific by querying for orders since January 1, 1998, type:

    ```
    select * from sample.orders where
    o_orderdate > '1998-01-01'
    ```

    Now only 129 records are returned.

11. The two back-end data sources have now been queried separately. To run a query to see if any orders were placed on the same date in both data sources, type:

    ```
    select DISTINCT sales.ord_num, sales.ord_date,
    orders.o_orderkey, orders.o_orderdate from sales,
    ```

```
sample.orders WHERE
sales.ord_date=orders.o_orderdate
```

21 records are returned, confirming that 21 orders were made on the same date in both data sources (DISTINCT ensures that only unique records are returned).

The two data sources are unrelated, so there is little else that can be queried on them heterogeneously. In reality, your back-end data sources will probably have more in common and you will be able to construct detailed SQL statements to select exactly the data you want.

12. Close iSQL unless you want to continue typing further SQL queries.

NB

You can query a virtual data source from within Microsoft Excel by using the **Data > Get External Data > New Database Query** command, which displays the **Choose Data Source** dialog box. Select the virtual data source that you want to query and then click **OK**. At this point, an error message saying that the selected data source contains 'no visible tables' is sometimes displayed.

To fix this, click **Options** on the **Query Wizard** screen to display the **Table Options** dialog box. Turn the **System Tables** option OFF if it is currently ON, or ON if it is currently OFF, and press **OK**. The tables in your virtual data source are then listed on the **Query Wizard** screen.

TECHNICAL REFERENCE

A

ODBC and SQL conformance

This section sets out the ODBC and SQL conformance for the Easysoft ODBC-SQI SQL Engine, listing the ODBC API function calls and the SQL components that are supported.

Appendix Guide

Introduction

Refer to this section if you are using the Easysoft ODBC-SQI SQL Engine in conjunction with (for example):

- your own ODBC calls for selecting or manipulating data
- complex SQL queries from within a commercial ODBC-compliant application, such as Microsoft Access.

This section *does not* describe ODBC API calls or the different components of SQL.

If you require more detailed information about implementing ODBC and SQL, the following publications may be of interest to you:

- *Microsoft ODBC 3.0 Programmer's Reference Volumes 1 and 2* (Microsoft Press, 1997)
- *International Standard for the Database Language (SQL)*, ISO/IEC 9075:1992
- X/Open CAE Specification, *SQL Call Level Interface (CLI), C451* (X/Open Company Ltd, UK, 1995)

For more general information about ODBC and SQL, please refer to the many reference books available on these subjects.

ODBC API Function Calls

ODBC API function calls can be used by programmers to enable applications to connect to data in databases.

By including an SQL statement as an argument of an ODBC function call, programmers can also develop features that allow end users to manipulate data in their databases.

NB The Purpose column in the following tables is taken from the *Microsoft ODBC 3.0 Programmer's Reference Volumes 1 and 2* (Microsoft Press, 1997).

The Easysoft ODBC-SQI SQL Engine provides the following ODBC API function calls:

Function	ODBC Conformance	Purpose
SQLAllocHandle	Core	Obtains an environment, connection, statement or descriptor handle.
SQLBindCol	Core	Assigns storage for a result column and specifies the data type.
SQLBindParameter	Core	Assigns storage for a parameter in an SQL statement.
SQLBrowseConnect	Level 1	Returns successive levels of connection attributes and valid attribute values. When a value has been specified for each connection attribute, connects to the data source.
SQLCancel	Core	Cancels an SQL statement.
SQLCloseCursor	Core	Closes a cursor that has been opened on a statement handle.
SQLColAttribute	Core	Describes attributes of a column in the results set.

Function	ODBC Conformance	Purpose
SQLColumnPrivileges	Level 2	Returns a list of columns and associated privileges for one or more tables.
SQLColumns	Core	Returns the list of column names in specified tables.
SQLConnect	Core	Connects to a specific driver by data source name, user ID and password.
SQLCopyDesc	Core	Copies descriptor information from one descriptor handle to another.
SQLDescribeCol	Core	Describes a column in the result set.
SQLDescribeParam	Level 2	Returns the description for a specific parameter in a statement.
SQLDisconnect	Core	Closes the connection.
SQLDriverConnect	Core	Connects to a specific driver by connection string or requests that the Driver Manager and driver display connection dialog boxes for the user.
SQLEndTran	Core	Commits or rolls back a transaction.
SQLExecDirect	Core	Executes a statement.
SQLExecute	Core	Executes a prepared statement.

Function	ODBC Conformance	Purpose
SQLFetch	Core	Returns multiple result rows.
SQLFetchScroll	Core	Returns scrollable result rows.
SQLForeignKeys	Level 2	Returns a list of column names that make up foreign keys, if they exist for a specified table.
SQLFreeHandle	Core	Releases an environment, connection, statement or descriptor handle.
SQLFreeStmt	Core	Ends statement processing, discards pending results, and, optimally, frees all resources associated with the statement handle.
SQLGetConnectAttr	Core	Returns the value of a connection attribute
SQLGetCursorName	Core	Returns the cursor name associated with a statement handle.
SQLGetData	Core	Returns part or all of one column of one row of a result set (useful for long data values).
SQLGetDescField	Core	Returns the value of a single descriptor field.
SQLGetDescRec	Core	Returns the values of multiple descriptor fields

Function	ODBC Conformance	Purpose
SQLGetDiagField	Core	Returns additional diagnostic information (a single field of the diagnostic data structure).
SQLGetDiagRec	Core	Returns additional diagnostic information (multiple fields of the diagnostic data structure).
SQLGetEnvAttr	Core	Returns the value of an environment variable.
SQLGetFunctions	Core	Returns supported driver functions
SQLGetInfo	Core	Returns information about a specific driver and data source
SQLGetStmtAttr	Core	Returns the value of a statement attribute.
SQLGetTypeInfo	Core	Returns information about supported data types.
SQLMoreResults	Level 1	Determines whether there are more result sets available and, if so, initializes processing for the next result set.
SQLNativeSql	Core	Returns the text of an SQL statement as translated by the driver.
SQLNumParams	Core	Returns the number of parameters in a statement.

Function	ODBC Conformance	Purpose
SQLNumResultCols	Core	Returns the number of columns in the result set.
SQLParamData	Core	Used in conjunction with SQLPutData to supply parameter data at execution time. (Useful for long data values.)
SQLPrepare	Core	Prepares and SQL statement for later execution.
SQLPrimaryKeys	Level 1	Returns the list of column names that make up the primary key for a table.
SQLProcedureColumns	Level 1	Returns the list of input and output parameters, as well as the columns that make up the result set for the specified procedures.
SQLProcedures	Level 1	Returns the list of procedure names stored in a specific data source.
SQLPutData	Core	Sends part or all of a data value for a parameter. (Useful for long data values.)
SQLRowCount	Core	Retuns the number of rows affected by an insert, update or delete request.
SQLSetConnectAttr	Core	Sets a connection attribute
SQLSetCursorName	Core	Specifies a cursor name.

Function	ODBC Conformance	Purpose
SQLSetDescField	Core	Sets a single descriptor field.
SQLSetDescRec	Core	Sets multiple descriptor fields.
SQLSetEnvAttr	Core	Sets an environment variable.
SQLSetPos	Level 1	Positions a cursor within a fetched block of data, and allows an application to refresh data in the rowset, or update or delete data in the result set.
SQLSetStmtAttr	Core	Sets a statement attribute.
SQLSpecialColumns	Core	Returns information about the optimal set of columns that uniquely identifies a row in a specified table, or the columns that are automatically updated when any value in the row is updated by a transaction.
SQLStatistics	Core	Returns statistics about a single table and the list of indexes associated with the table.
SQLTablePrivileges	Level 2	Returns a list of tables and the privileges associated with each table.

Function	ODBC Conformance	Purpose
SQLTables	Core	Returns the list of table names stored in a specific data source.

Figure 15: ODBC API functions

The following functions are provided by the ODBC Driver Manager:

Function	ODBC Conformance	Purpose
SQLDataSources	Core	Returns the list of available data sources
SQLDrivers	Core	Returns the list of installed drivers and their attributes

Figure 16: ODBC Driver Manager functions

The following functions are provided by the Setup DLL:

Function	ODBC Conformance	Purpose
SQLConfigDriver	Core	Loads the driver-specific setup DLL.
SQLConfigDSN	Core	Adds, modifies or deletes a data source.

Figure 17: ODBC Setup DLL functions

The following functions have been superceded, but are still supported by theEasysoft ODBC-SQI SQL Engine:

Function	ODBC Conformance	Purpose
SQLAllocConnect	Core	ODBC 2.x function superseded by SQLAllocHandle
SQLAllocEnv	Core	ODBC 2.x function superseded by SQLAllocHandle
SQLError	Core	ODBC 2.x function superseded by SQLGetDiagRec
SQLGetConnectOption	Core	ODBC 2.x function superseded by SQLGetConnectAttr
SQLSetParam	Core	ODBC 1 function, superseded by SQLBindParameter

Figure 18: Superceeded functions

The following function is currently not supported, but are planned for future release:

Function	ODBC Conformance	Purpose
SQLBulkOperations	Level 1	Performs bulk insertions and bulk bookmark operations, including update, delete and fetch by bookmark.

Figure 19: Unsuppported functions

Statement Types

The Easysoft ODBC-SQI SQL Engine supports the following SQL statements:

SQL Minimum Grammar	Additional
CREATE TABLE	ALTER TABLE
	CREATE INDEX
	CREATE VIEW
DELETE STATEMENT (searched)	DELETE (positioned)
DROP TABLE	DROP INDEX
	DROP VIEW
INSERT	
SELECT	SELECT FOR UPDATE
UPDATE (searched)	UPDATE (positioned)
	GRANT
	REVOKE
	COMMIT
	ROLLBACK

Figure 20: Supported statements

NB	Some of these commands (e.g. COMMIT and ROLLBACK) are only available if they are supported by the back-end DBMS. Also, unless you are using link names, you can only use the CREATE, ALTER and DROP commands (the SQL Data Definition Language commands) if the virtual data source has only one DBMS at the 'back-end'.

Table References

The Table reference list in a SELECT query can contain all or any of:

- Table name
- Subquery
- Join

Constructs

The Easysoft ODBC-SQI SQL Engine supports the following constructs:

INNER JOIN

LEFT OUTER JOIN

RIGHT OUTER JOIN

FULL OUTER JOIN

NATURAL JOIN

UNION

UNION ALL

Correlated and non-correlated subqueries

TRIM, SUBSTRING, CHARACTER_LENGTH, BIT_LENGTH, OCTET_LENGTH and POSITION value functions

The SQL-92 CAST function and the ODBC CONVERT function for conversion between compatible data types.

When using JOINs, note that:

- The joining condition may be specified with the ON or USING clause.
- Both the left and right source can be a table name, subquery or another join.
- Joins or subqueries can be nested with no restriction on depth.
- Joins can be specified in both SQL92 and ODBC format. For example:

  SQL

  ```
  SELECT * from x LEFT OUTER JOIN y ON x.a =
  y.a
  ```

  ODBC

  ```
  SELECT * from {oj x LEFT OUTER y ON x.a =
  y.a }
  ```

Predicates

The Easysoft ODBC-SQI SQL Engine supports the following predicates:

Predicate	Example Syntax
Comparisons, = <> < <= > >=	WHERE *x* = *y*
BETWEEN	WHERE *a* BETWEEN *x* AND *y*
LIKE / NOT LIKE	WHERE *a* LIKE '%green%'
NULL / NOT NULL	WHERE *a* is NULL
IN value_list	WHERE *a* IN (*value1*, *value2*, *value3*)
IN sub_query	WHERE *a* IN (SELECT *x* FROM *y*)
Quantified comparison	WHERE *a* = SOME (SELECT *x* FROM *y*)
EXISTS subquery	WHERE EXISTS (SELECT *x* FROM *y*)
CASE, NULLIF, COALESCE	Refer to an SQL reference for example syntax of these conditional expressions.

Figure 21: Supported predicates

Subqueries in predicates can be correlated or non-correlated:

Correlated

```
SELECT a FROM b WHERE c = ALL ( SELECT x FROM y
WHERE z = a )
```

Non-correlated

```
SELECT a FROM b WHERE c = ALL ( SELECT x FROM y
WHERE z = 12 )
```

Scalar Functions

The Easysoft ODBC-SQI SQL Engine provides all the functions required by ODBC and also functions from SQL92.

Functions can be specified in SQL92 or ODBC format.

For example:

SQL

```
SELECT CURRENT_DATE, EXTRACT( YEAR FROM
 Employee.data_of_birth ) FROM Employee
```

ODBC

```
SELECT {fn CURRENT_DATE()}, {fn EXTRACT( YEAR
 FROM Employee.data_of_birth )} FROM Employee
```

NB The Description column in the following tables is taken from the *Microsoft ODBC 3.0 Programmer's Reference*.

The following functions are supported by the Easysoft ODBC-SQI SQL Engine:

String Functions

Function	Description
ASCII(*string_exp*)	Returns the ASCII code value of the leftmost character of *string_exp* as an integer.
BIT_LENGTH(*string_exp*)	Returns the length in bits of the string expression.

Function	Description
CHAR(*code*)	Returns the character that has the ASCII code value specified by *code*. The value of *code* should be between 0 and 255, otherwise the return value is data source-dependent.
CHAR_LENGTH(*string_exp*)	Returns the length in characters of the string expression, if the string expression is of a character data type, otherwise returns the length in bytes of the string expression (the smallest integer not less than the number of bits divided by 8). This is the same function as CHARACTER_LENGTH.
CHARACTER_LENGTH	See CHAR_LENGTH
CONCAT(*string_exp1, string_exp2*)	Returns a character string that is the result of concatenating *string_exp2* to *string_exp1*. The resulting string is DBMS-dependent.
DIFFERENCE(*string_exp1, string_exp2*)	Returns an integer value that indicates the difference between the values returned by the SOUNDEX function for *string_exp1* and *string_exp2*.
INSERT(*string_exp1, start, length, string_exp2*)	Returns a character string where *length* characters have been deleted from *string_exp1* beginning at *start* and where *string_exp2* has been inserted into *string_exp*, beginning at *start*.
LCASE(*string_exp*)	Returns a string equal to that in *string_exp* with all uppercase characters converted to lowercase.
LEFT(*string_exp*, count)	Returns the leftmost *count* characters of *string_exp*.
LENGTH(*string_exp*)	Returns the number of characters in *string_exp*, excluding trailing blanks.

Function	Description
LOCATE(*string_exp1*, *string_exp2*[,*start*])	Returns the starting position of the first occurrence of *string_exp1* within *string_exp2*. The search for the first occurrence of *string_exp1* begins with the first character position in *string_exp2* unless the optional argument, *start*, is specified. If *start* is specified, the search begins with the character position indicated by the value of *start*. The first character position in *string_exp2* is indicated by the value1. If *string_exp1* is not found within *string_exp2*, the value 0 is returned.
LTRIM(*string_exp*)	Returns the characters of *string_exp*, with leading blanks removed.
OCTET_LENGTH(*string_exp*)	Returns the length in bytes of the string expression. The result is the smallest integer not less than the number of bits divided by 8.
POSITION(*char_exp* IN *char_exp*)	Returns the position of the first character expression in the second character expression. The result is an exact numeric with an implementation-defined precision and a scale of 0.
REPEAT(*string_exp*, *count*)	Returns a character string composed of *string_exp* repeated *count* times.
REPLACE(*string_exp1*, *string_exp2*, *string_exp3*)	Search *string_exp1* for occurrences of *string_exp2* and replace with *string_exp3*.
RIGHT(*string_exp*, *count*)	Returns the rightmost *count* characters of *string_exp*.
RTRIM(*string_exp*)	Returns the characters of *string_exp* with trailing blanks removed.
SOUNDEX(*string_exp*)	Returns a data source-dependent character string representing the sound of the words in *string_exp*.

Function	Description
SPACE(*count*)	Returns a character string consisting of *count* spaces.
SUBSTRING(*string_exp*, *start*, *length*)	Returns a character string that is derived from *string_exp* beginning at the character position specified by *start* for *length* characters.
TRIM	This is an SQL92 version of the ODBC LTRIM/RTRIM functions.
UCASE(*string_exp*)	Returns a string equal to that in *string_exp* with all lowercase characters converted to uppercase.

Figure 22: Supported string functions

NUMERIC FUNCTIONS

Function	Description
ABS(*numeric_exp*)	Returns the absolute value of *numeric_exp*.
ACOS(*float_exp*)	Returns the arccosine of *float_exp* as an angle, expressed in radians.
ASIN(*float_exp*)	Returns the arcsine of *float_exp* as an angle, expressed in radians.
ATAN(*float_exp*)	Returns the arctangent of *float_exp* as an angle, expressed in radians.
ATAN2(*float_exp1, float_exp2*)	Returns the arctangent of the x and y coordinates, specified by *float_exp1* and *float_exp2* respectively, as an angle expressed in radians.
CEILING(*numeric_exp*)	Returns the smallest integer greater than or equal to *numeric_exp*.

Function	Description
COS(*float_exp*)	Returns the cosine of *float_exp* where *float_exp* is an angle expressed in radians.
COT(*float_exp*)	Returns the cotangent of *float_exp* where *float_exp* is an angle expressed in radians.
DEGREES(*numeric_exp*)	Returns the number of degrees converted from *numeric_exp* radians.
EXP(*float_exp*)	Returns the exponential value of *float_exp*.
FLOOR(*numeric_exp*)	Returns the largest integer less than or equal to *numeric_exp*.
LOG(*float_exp*)	Returns the natural logarithm of *float_exp*.
LOG10(*float_exp*)	Returns the base 10 logarithm of *float_exp*.
MOD(*integer_exp1*, *integer_exp2*)	Returns the remainder (modulus) of integer_exp1 divided by *integer_exp2*.
PI()	Returns the constant value of pi as a floating point value.
POWER(*numeric_exp*, *integer_exp*)	Returns the value of *numeric_exp* to the power of *integer_exp*.
RADIANS(*numeric_exp*)	Returns the number of radians converted from *numeric_exp* degrees.
RAND([*integer_exp*])	Returns a random floating point value using *integer_exp* as the optional seed value.
ROUND(*numeric_exp*, *integer_exp*)	Returns *numeric_exp* rounded to *integer_exp* places right of the decimal point. If *integer_exp* is negative, *numeric_exp* is rounded to |*integer_exp*| places to the left of the decimal point.
SIGN(*numeric_exp*)	Returns an indicator of the sign of *numeric_exp*. If *numeric_exp* is less than zero, -1 is returned. If numeric_exp equals aero, 0 is returned. If *numeric_exp* is greater than zero, 1 is returned.

Function	Description
SIN(*float_exp*)	Returns the sine of *float_exp*, where *float_exp* is an angle expressed in radians.
SQRT(*float_exp*)	Returns the square root of *float_exp*.
TAN(*float_exp*)	Returns the tangent of *float_exp* where *float_exp* is an angle expressed in radians.
TRUNCATE(*numeric_exp*, *integer_exp*)	Returns *numeric_exp* truncated to *integer_exp* places right of the decimal point. If *integer_exp* is negative, *numeric_exp* is truncated to |*integer_exp*| places to the left of the decimal point.

Figure 23: Supported numeric functions

TIME, DATE AND INTERVAL FUNCTIONS

Function	Description
CURRENT_DATE()	Returns the current date.
CURRENT_TIME[(*time-precision*)]	Returns the current local time. The *time-precision* argument determines the seconds precision of the returned value.
CURRENT_TIMESTAMP[(*timestamp-precision*)]	Returns the current local date and local time as a timestamp value. The *timestamp-precision* argument determines the seconds precision of the returned timestamp.
CURDATE()	Returns the current date.
CURTIME()	Returns the current local time.
DAYNAME(*date_exp*)	Returns a character string containing the data source-specific name of the day for the day portion of *date_exp*.

Function	Description
DAYOFMONTH(*date_exp*)	Returns the day of the month based on the month field in *date_exp* as an integer value in the range 1-31.
DAYOFWEEK(*date_exp*)	Returns the day of the week based on the week field in *date_exp* as an integer value in the range of 1-7 where 1 represents Sunday.
DAYOFYEAR(*date_exp*)	Returns the day of the year based on the year field in *date_exp* as an integer value in the range of 1-366.
EXTRACT(*extract-field FROM extract-source*)	Returns the *extract-field* portion of the *extract-source*. The *extract-source* argument is a datetime or interval expression. The *extract-field* argument can be one of the YEAR, MONTH, DAY, HOUR, MINUTE, SECOND keywords. The precision of the returned value is implementation-defined. The scale is 0 unless SECOND is specified, in which case the scale is not less that the fractional seconds precision of the *extract-source* field.
HOUR(*time_exp*)	Returns the hour based on the hour field in *time_exp* as an integer value in the range of 0-23.
MINUTE(*time_exp*)	Returns the minute based on the minute field in *time_exp* as an integer value in the range of 0-59.
MONTH(*date_exp*)	Returns the month based on the month field in *date_exp* as an integer value in the range 1-12.
MONTHNAME (*date_exp*)	Returns a character string containing the data source-specific name of the month for the month portion of *date_exp*.
NOW()	Returns the current date and time as a timestamp value.

Function	Description
QUARTER(*date_exp*)	Returns the quarter in *date_exp* as an integer value in the range of 1-4.
SECOND(*time_exp*)	Returns the second based on the second field in *time_exp* as an integer value in the range of 0-59.
TIMESTAMPADD (*interval*, *integer_exp*, *timestamp_exp*)	Returns the timestamp calculated by adding *integer_exp* intervals of type *interval* to *timestamp_exp*. Valid values of *interval* are the following keywords: SQL_TSI_FRAC_SECOND, SQL_TSI_SECOND, SQL_TSI_MINUTE, SQL_TSI_HOUR, SQL_TSI_DAY, SQL_TSI_WEEK, SQL_TSI_MONTH, SQL_TSI_QUARTER, SQL_TSI_YEAR where fractional seconds are expressed in billionths of a second. If *time_stamp* is a time value and *interval* specifies days, weeks, months, quarters, or years, the date portion of *timestamp_exp* is set to the current date before calculating the resulting timestamp. If *timestamp_exp* is a date value and *interval* specifies fractional seconds, seconds, minutes, or hours, the time portion of *timestamp_exp* is set to 0 before calculating the resulting timestamp. An application determines which intervals a data source supports by calling SQLGetInfo with the SQL_TIMEDATE_ADD_INTERVALS option.

Function	Description
TIMESTAMPDIFF (*interval*, *timestamp_exp1*, *timestamp_exp2*)	Returns the integer number of intervals of type *interval* by which *timestamp_exp2* is greater than *timestamp_exp1*. The keywords SQL_TSI_FRAC_SECOND, SQL_TSI_SECOND, SQL_TSI_MINUTE, SQL_TSI_HOUR, SQL_TSI_DAY, SQL_TSI_WEEK, SQL_TSI_MONTH, SQL_TSI_QUARTER, SQL_TSI_YEAR are valid values of *interval*, where fractional seconds are expressed in billionths of a second. If either timestamp expression is a time value and *interval* specifies days, weeks, months, quarters, or years, the date portion of that timestamp is set to the current date before calculating the difference between timestamps. If either timestamp expression is a date value and *interval* specifies fractional seconds, seconds, minutes, or hours, the time portion of that timestamp is set to 0 before calculating the difference between timestamps. An application determines which intervals a data source supports by calling SQLGetInfo with the SQL_TIMEDATE_DIFF_INTERVALS option.
WEEK(*date_exp*)	Returns the week of the year based on the week field in *date_exp* as an integer value in the range of 1-53.
YEAR(*date_exp*)	Returns the year based on the year field in *date_exp* as an integer value. The range is data source-dependent.

Figure 24: Supported time, date and interval functions

System Functions

Function	Description
DATABASE()	Returns the name of the database corresponding to the connection handle.
IFNULL(*exp*, *value*)	If *exp* is null, *value* is returned. If *exp* is not null, *exp* is returned. The possible data type or types of *value* must be compatible with the data type of *exp*.
USER()	Returns the user name in the DBMS. This may be different from the login name.
CURRENT_USER	This is an SQL92 version of the ODBC USER function.

Figure 25: Supported system functions

Set Functions

The Easysoft ODBC-SQI SQL Engine supports the following set functions:

COUNT

AVG

MIN

MAX

SUM

For example, the following query would return the maximum price from the PRODUCT table:

```
SELECT MAX (PRICE) FROM PRODUCT
```

NB	If your ODBC application does not return any results when using a set function, try inserting an AS clause into your query. For example: `SELECT MAX (PRICE) AS COST FROM PRODUCT`

Data Types

The following SQL data types are supported, as described in Appendix D of the *Microsoft ODBC 3.0 Programmer's Reference*.

SQL_CHAR

SQL_VARCHAR

SQL_LONGVARCHAR

SQL_DECIMAL

SQL_NUMERIC

SQL_SMALLINT

SQL_INTEGER

SQL_REAL

SQL_FLOAT

SQL_DOUBLE

SQL_BIT

SQL_TINYINT

SQL_BIGINT

SQL_BINARY

SQL_VARBINARY

SQL_LONGVARBINARY

SQL_TYPE_DATE

SQL_TYPE_TIME

SQL_TYPE_TIMESTAMP

SQL_INTERVALS (all types)

Literals

All SQL92 and ODBC literals are supported and can be specified in either form. For example:

SQL92

```
DATE '1999-01-02', INTERVAL '10-2' YEAR TO MONTH
```

ODBC

```
{d '1999-01-02'}, {INTERVAL '10-2' YEAR TO MONTH}
```

Data Type Conversions

The Easysoft ODBC-SQI SQL Engine supports both the SQL92 CAST function and the ODBC CONVERT FUNCTION for conversion between compatible data types.

The following table shows how compatible data types are converted.

NB For conciseness, the SQL_ prefix has been omitted from all data types in the Supported data type conversions table.

	CHAR	VAR CHAR	LONG VAR CHAR	DECIMAL	NUMERIC	BIT	TINY INT	SMALL INT	INTEGER	BIG INT	REAL	FLOAT	DOUBLE	BINARY	VAR BINARY	LONG VAR BINARY	DATE	TIME	TIME STAMP	INTERVAL _YEAR	INTERVAL _DAY
CHAR	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
VARCHAR	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
LONGVARCHAR	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
DECIMAL	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N
NUMERIC	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N
BIT	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N
TINYINT	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N
SMALLINT	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N
INTEGER	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N
BIGINT	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N
REAL	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N
FLOAT	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N
DOUBLE	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N
BINARY	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
VARBINARY	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
LONGVARBINARY	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
DATE	Y	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	Y	N	N
TIME	Y	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	N
TIMESTAMP	Y	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	Y	Y	N	N
INTERVAL_YEAR	Y	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
INTERVAL_DAY	Y	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Figure 26: Supported data type conversions

Numeric Data Types

The following table shows which numeric data type is adopted when arithmetic operations (+-*/) are performed on two numerics of different data types.

+	DECIMAL	NUMERIC	SMALLINT	INTEGER	TINYINT	BIGINT	REAL	FLOAT	DOUBLE	BIT
DECIMAL	DECIMAL	NUMERIC	DECIMAL	DECIMAL	DECIMAL	DECIMAL	DECIMAL	DECIMAL	DECIMAL	DECIMAL
NUMERIC	NUMERIC	NUMERIC	NUMERIC	NUMERIC	NUMERIC	NUMERIC	NUMERIC	NUMERIC	NUMERIC	NUMERIC
SMALLINT	DECIMAL	NUMERIC	SMALLINT	INTEGER	SMALLINT	BIGINT	REAL	FLOAT	DOUBLE	SMALLINT
INTEGER	DECIMAL	NUMERIC	INTEGER	INTEGER	INTEGER	BIGINT	REAL	FLOAT	DOUBLE	INTEGER
TINYINT	DECIMAL	NUMERIC	SMALLINT	INTEGER	TINYINT	BIGINT	REAL	FLOAT	DOUBLE	TINYINT
BIGINT	DECIMAL	NUMERIC	BIGINT	BIGINT	BIGINT	BIGINT	BIGINT	BIGINT	BIGINT	BIGINT
REAL	DECIMAL	NUMERIC	REAL	REAL	REAL	BIGINT	REAL	FLOAT	DOUBLE	REAL
FLOAT	DECIMAL	NUMERIC	FLOAT	FLOAT	FLOAT	BIGINT	FLOAT	FLOAT	DOUBLE	FLOAT
DOUBLE	DECIMAL	NUMERIC	DOUBLE	DOUBLE	DOUBLE	BIGINT	DOUBLE	DOUBLE	DOUBLE	DOUBLE
BIT	DECIMAL	NUMERIC	SMALLINT	INTEGER	TINYINT	BIGINT	DOUBLE	DOUBLE	DOUBLE	BIT

Figure 27: Adopted numeric data types

String Concatenation

A string data type (SQL_CHAR, SQL_VARCHAR) can be concatenated with another string data type using the concat operator (|| or +).

If either of the operands are of variable length, then the result will also be of varying length.

Interval Operations

When adding or subtracting an integral numeric value to or from an interval value where the interval value is a single field, the interval type does not change.

For example, if you have the interval value 3 DAY and you add 6, the result will be 9 DAY - the interval is not recalculated as a number of weeks and days.

Optimization

The Easysoft ODBC-SQI SQL Engine performs several optimizations to improve performance, including query and table ordering optimizations:

Query Optimization

The WHERE clause of a query will be rewritten into a form that allows more efficient processing of data. For example the query:

```
SELECT * FROM x WHERE ( a = 10 or b = 20 ) and c = 30
```

will be rewritten as the equivalent:

```
SELECT * FROM x WHERE a = 10 and c = 30

UNION

SELECT * FROM x WHERE b = 20 and c = 30 and a <> 10
```

Table Optimization

In cases where indexes are present on tables, the Easysoft ODBC-SQI SQL Engine will, if necessary, rearrange the sequence in which tables are processed in order to enable an index to be used.

This will minimize the number of reads and positions executed and lead to huge increases in performance.

Informational Schema

The Easysoft ODBC-SQI SQL Engine exposes an informational schema view of the tables supplied by the target data sources, which consists of the following system tables:

INFO_SCHEMA.CHARACTER_SETS

INFO_SCHEMA.COLLATIONS

INFO_SCHEMA.COLUMN_PRIVILEGES

INFO_SCHEMA.COLUMNS

INFO_SCHEMA.INDEXES

INFO_SCHEMA.SCHEMATA

INFO_SCHEMA.SERVER_INFO

INFO_SCHEMA.SQL_LANGUAGES

INFO_SCHEMA.TABLE_PRIVILEGES

INFO_SCHEMA.TABLES

INFO_SCHEMA.USAGE_PRIVILEGES

INFO_SCHEMA.VIEWS

To get information about all the tables in the data source, use:

```
SELECT * FROM INFO_SCHEMA.TABLES
```

To get information about all the columns in the data source, use:

```
SELECT * FROM INFO_SCHEMA_COLUMNS
```

To get information about all the columns in a specific table, use:

```
SELECT * FROM INFO_SCHEMA_COLUMNS WHERE
 TABLE_NAME='table_name'
```

ODBC Features

CURSORS

The Easysoft ODBC-SQI SQL Engine supports FORWARD ONLY, STATIC and KEYSET cursors (but not DYNAMIC cursors).

The Easysoft ODBC-SQI SQL Engine supports:

- CONCURRENCY_LOCK and CONCURRENCY_VALUE cursor concurrency
- Row and column binding
- Rowset sizes > 1
- Bookmarks

ASYNCHRONOUS OPERATION

The ODBC Programmer's Reference lists the functions that can be executed asynchronously. The Easysoft ODBC-SQI SQL Engine supports asynchronous operation for these functions although SQL_STILL_EXECUTING will not be returned for a function which happens immediately because of an earlier function. For example, SQLGetData will always return immediately whereas SQLExecute may not.

For example, the following query would probably not execute immediately:

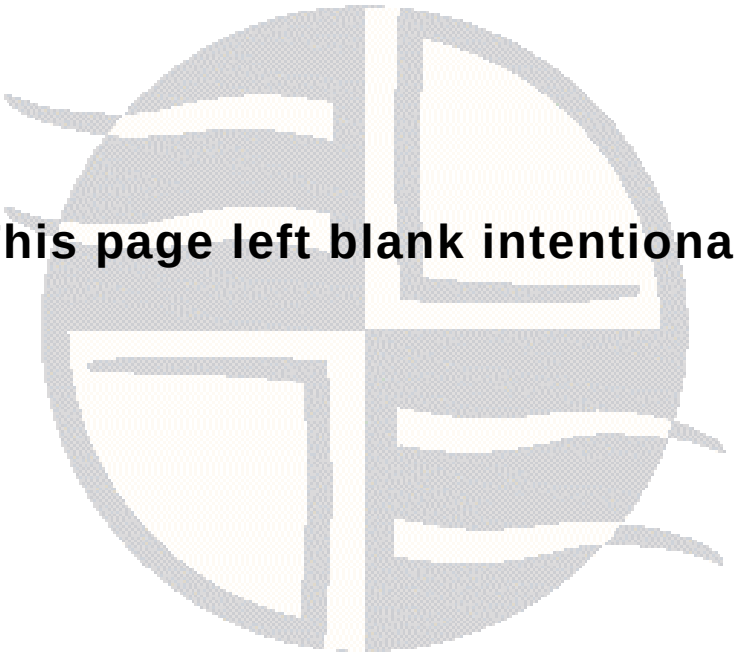
```
SELECT * FROM EMPLOYEE

WHERE SALARY > 20000

ORDER BY AGE
```

whereas this query probably would:

```
SELECT * FROM EMPLOYEE WHERE UNIQUE_ID = 101
```



This page left blank intentionally

GLOSSARY

B

Terms and definitions

API (Application Programmer Interface)

A published set of function calls and constants allowing different programmers to utilize a ready-written library of subroutines.

Application

A program that applies the computer to solving some real-world problem. In ODBC terms, it is the program connecting to the data source.

Authorization code

You must have an authorization code for the Easysoft product you wish to license in order to obtain a purchased license. When you purchase a product your authorization code is emailed to you. You do not need an authorization code to obtain a trial license.

Back-end data source

A term used in this manual to refer to the individual ODBC data sources linked into a virtual data source.

Back-end DBMS

A term used in this manual to refer to the native DBMS of back-end data sources, such as Oracle or Microsoft SQL Server.

Client

A process performing tasks local to the current user, such as formatting and displaying a report from data retrieved from the server.

Client server

The architecture whereby one process (the server) keeps track of global data, and another task (the client) is responsible for formatting and presenting the data. The client connects to the server and requests queries or actions be performed on its behalf. Often these processes run on different hosts across a local-area network.

Column

The vertical dimension of a table. Columns are named and have a domain (or type).

Database

A collection of data files.

Data source

In ODBC terms, a data source is a database or other data repository coupled with an ODBC Driver, which has been given a Data Source Name (see **“DSN” on page 95**) to identify it to the ODBC Driver Manager.

Data type

The specification of permitted values. A data type limits the values which are allowed to be used.

DBMS

Database Management System - software that handles access to a database.

Driver Manager

Software whose main function is to load ODBC drivers. ODBC applications connect to the Driver Manager and request a data source name (DSN). The Driver Manager loads the driver specified

in the DSN's configuration file. See also **“ODBC driver” on page 95**.

DSN

Data Source Name. A name associated with an ODBC data source. Driver Managers use the Data Source Name to cross-reference configuration information and load the required driver.

Field

A placeholder for a single datum in a record, for example you can have a Surname field in a Contact Details record. Fields are sometimes referred to as cells.

FTP

File Transfer Protocol. A standard method of transferring files between different machines.

Middleware

Software that is placed between the client and the server to improve or expand functionality.

License key

A string which is provided by Easysoft for use in the licensing process.

ODBC

Open Data Base Connectivity. A programming interface that enables applications to access data in database management systems that use Structured Query Language (SQL) as a data access standard.

ODBC driver

Software that accesses a proprietary data source, providing a standardized view of the data to an ODBC-compliant application.

Row

The horizontal dimension of a table. At its most basic, a row might equate to a record within a file.

Schema

A specification of the structure of a database, including the tables, their column headings and keys.

Server

A computer, or host, on the network, designed for power and robustness rather than user-friendliness and convenience. Servers typically run around-the-clock and carry central corporate data.

OR

A process performing the centralized component of some task, for example, extracting information from a corporate database.

SQL

Structured Query Language. An international standard text language for querying and manipulating databases.

System data source

A data source which can be accessed by any user on a given system. See also **“User data source” on page 97**.

Table

A data set in a relational database, composed of rows and columns.

TCP/IP

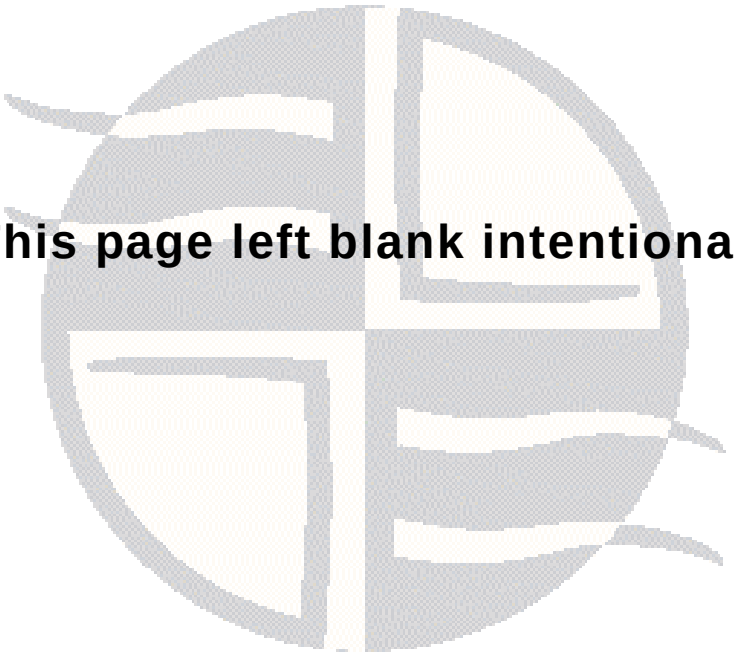
Transmission Control Protocol/Internet Protocol. A standard method of transferring data between different machines.

User data source

A data source which can only be accessed by a specific user on a given system. See also **“System data source” on page 96**.

Virtual data source

A data source configured to connect to multiple local or remote data sources, providing heterogeneous access to the data and providing ODBC 3.5 functionality, even if the existing ODBC drivers conform to lower levels.



This page left blank intentionally

INDEX

F

I

J

L

M

N

O

P

R

S

T

U

V

W