Easysoft ODBC-Access Driver User's Guide

This manual documents version 1.4.n of the Easysoft ODBC-Access Driver.

Table of contents

Getting started . . . 4
Installing the Easysoft ODBC-Access Driver . . . 5
Installing on Linux or UNIX . . . 5
Uninstalling on Linux or UNIX . . . 13
Connecting to Access . . . 14
Connecting from Linux or UNIX . . . 14
Connection attributes . . . 15
Connection examples . . . 21
Troubleshooting database connection problems . . . 25
DSN-less connections . . . 31
Logging . . . 32
ODBC Driver Manager logging on Linux or UNIX . . . 32
Easysoft ODBC-Access Driver logging on Linux and UNIX . . . 32
Client applications . . . 33
Oracle . . . 34
Connecting Access to Oracle on Linux and UNIX . . . 34
LibreOffice . . . 36
Go . . . 37
Node.js . . . 38
Perl . . . 40
PHP . . . 43
Python . . . 45
R . . . 47
About the Easysoft ODBC-Access Driver . . . 49
ODBC API and scalar functions . . . 50
API functions . . . 50
Scalar functions . . . 52
Data type mapping . . . 55
SQL examples . . . 56
Example queries . . . 56
Example inserts, updates, and deletes . . . 62
Index . . . 65

Getting started

This section shows you how to install the Easysoft ODBC-Access Driver and configure the ODBC data source that stores the connection details for your Access database. You're then ready to work with Access data in your application.

- Installing the Easysoft ODBC-Access Driver
- Connecting to Access
- Logging

Installing the Easysoft ODBC-Access Driver

Install the Easysoft ODBC-Access Driver on the computer where the application you want to connect to Access is running.

Installing on Linux or UNIX

The installation can be done by anyone with root access.

1. Download the Easysoft ODBC-Access Driver distribution for your client application platform.

   If your client application is 64-bit, choose the 64-bit driver distribution from the **Platforms** list. If your client application is 32-bit, choose the 32-bit driver distribution from the **Platforms** list.
2. Copy the distribution to a temporary directory on the machine where the application you want to connect to Access is installed.
3. Unpack the distribution and cd into the resultant directory.
4. As root, run:

   ```
   ./install
   ```
5. Follow the onscreen instructions to progress through the installation.

Further information

- Preinstallation requirements
- What you can install
- Where to install
- Changes made to your system
- Installing alongside other existing Easysoft product installations
- Gathering information required during the installation
- Unpacking the distribution
- License to use
- Answering questions during the installation
- Running the installer
- Locating or installing unixODBC
- Installing the Easysoft ODBC driver
- Licensing
- Post installation steps for non-root installations

Preinstallation requirements

To install the Easysoft ODBC-Access Driver you need:

- The Bourne shell in /bin/sh. If your Bourne shell is not located there, you may need to edit the first line of the installation script.
- Various commonly used commands such as:

  ```
  grep, awk, test, cut, ps, sed, cat, wc, uname, tr, find, echo, sum, head, tee, id
  ```

  If you do not have any of these commands, they can usually be obtained from the Free Software Foundation. As the tee command does not work correctly on some systems, the distribution includes a tee replacement.
- Depending on the platform, you'll need up to 10 MB of temporary space for the installation files and up to 10 MB of free disk space for the installed programs. If you also install the unixODBC Driver Manager, these numbers increase by approximately 1.5 MB.
- For Easysoft licensing to work, you must do one of the following:

  - Install the Easysoft ODBC-Access Driver in /usr/local/easysoft.
  - Install the Easysoft ODBC-Access Driver elsewhere and symbolically link /usr/local/easysoft to wherever you chose to install the software.

    The installation will do this automatically for you so long as you run the installation as someone with permission to create /usr/local/easysoft.

  - Install the Easysoft ODBC-Access Driver elsewhere and set the EASYSOFT_ROOT environment variable. For more information about setting the EASYSOFT_ROOT environment variable, refer to Post installation steps for non-root installations.
- An ODBC Driver Manager.

Easysoft ODBC-Access Driver distributions include the unixODBC Driver Manager.

- You do not have to be the root user to install, but you will need permission to create a directory in the chosen installation path. Also, if you are not the root user, it may not be possible for the installation to:
  1. Register the Easysoft ODBC-Access Driver with unixODBC.
  2. Create the example data source in the SYSTEM odbc.ini file.
  3. Update the dynamic linker entries (some platforms only).

If you are not root, these tasks will have to be done manually later.

We recommend that you install all components as the root user.

What you can install

This distribution contains:

- The Easysoft ODBC-Access Driver.
- The unixODBC Driver Manager.

You need an ODBC Driver Manager to use the Easysoft ODBC-Access Driver from your applications. The distribution therefore contains the unixODBC Driver Manager. Most (if not all) UNIX and Linux applications support the unixODBC Driver Manager. For example, Perl DBD::ODBC, PHP, Python, and so on.

You do not have to install the unixODBC Driver Manager included with this distribution. You can use an existing copy of unixODBC. For example, a version of unixODBC installed by another Easysoft product, a version obtained from your operating system vendor, or one that you built yourself. However, as Easysoft ensure that the unixODBC distributed with the Easysoft ODBC-Access Driver has been tested with that driver, we recommend you use it.

If you choose to use an existing unixODBC Driver Manager, the installation script will attempt to locate it. The installation script looks for the ODBC Driver Manager in the standard places. If you have installed it in a non-standard location, the installation script prompts you for the location. The installation primarily needs unixODBC's odbcinst command to install drivers and data sources.

Where to install

This installation needs a location for the installed files. The default location is /usr/local.

At the start of the installation, you're prompted for an installation path. All files are installed in a subdirectory of your specified path called easysoft. For example, if you accept the default location /usr/local, the product will be installed in /usr/local/easysoft and below.

If you choose a different installation path, the installation script tries to symbolically link /usr/local/easysoft to the easysoft subdirectory in your chosen location. This allows us to distribute binaries with built in dynamic linker run paths. If you are not root or the path /usr/local/easysoft already exists and is not a symbolic link, the installation will be unable to

create the symbolic link. For information about how to correct this manually, refer to Post installation steps for non-root installations.

Note that you cannot license Easysoft products until either of the following is true:

- /usr/local/easysoft exists either as a symbolic link to your chosen installation path or as the installation path itself.
- You have set EASYSOFT_ROOT to *installation_path*/easysoft.

Changes made to your system

The installation script installs files in subdirectories of the path requested at the start of the installation. Depending on what is installed, a few changes may be made to your system:

1. If you choose to install the Easysoft ODBC-Access Driver into unixODBC, unixODBC's odbcinst command will be run to add an entry to your odbcinst.ini file. You can locate this file with odbcinst -j. (odbcinst is in *installation_path*/easysoft/unixODBC/bin, if you are using the unixODBC included with this distribution.)
2. The installation script installs an example data source into unixODBC. This data source will be added to your SYSTEM odbc.ini file. You can locate your SYSTEM odbc.ini file by using odbcinst -j.
3. Dynamic linker. On operating systems where the dynamic linker has a file listing locations for shared objects (Linux and FreeBSD), the installation script will attempt to add paths under the path you provided at the start of the installation to the end of this list:
   - On Linux, this is usually the file /etc/ld.so.conf.
   - On FreeBSD this is usually the file /etc/defaults/rc.conf.

Installing alongside other existing Easysoft product installations

Each Easysoft distribution contains common files shared between Easysoft products. These shared objects are placed in *installation_path*/easysoft/lib. When you run the installation script, the dates and versions of these files are compared with the same files in the distribution. The files are only updated if the files being installed are newer or have a later version number.

You should ensure that nothing on your system is using Easysoft software before starting an installation. This is because on some platforms, files in use cannot be replaced. If a file cannot be updated, you get a warning during the installation. All warnings are written to a file called warnings in the directory you unpacked the distribution into.

If the installer detects you're upgrading a product, the installer will suggest you delete the product directory to avoid having problems with files in use. An alternative is to rename the specified directory.

If you are upgrading, you will need a new license from Easysoft to use the new driver.

Gathering information required during the installation

During the installation, you're prompted for various pieces of information. Before installing, you need to find out whether you have unixODBC already installed and where it is installed. The installation script searches standard places like /usr and /usr/local.

However, if you installed the Driver Manager in a non-standard place and you do not install the included unixODBC, you will need to know the location.

Unpacking the distribution

The distribution for UNIX and Linux platforms is a tar file. To extract the installation files from the tar file, use:

```
tar -xvf odbc-access-1.4.0-linux-x86-64-ul64.tar
```

This creates a directory with the same name as the tar file (without the .tar postfix) containing further archives, checksum files, an installation script, and various other installation files.

Change into the directory created by unpacking the tar file to run the installation script. For example:

```
# cd odbc-access-1.4.0-linux-x86-64-ul64
```

License to use

The end-user license agreement (EULA) is in the file license.txt. Be sure to understand the terms of the agreement before continuing, as you're required to accept the license terms at the start of the installation.

Answering questions during the installation

Throughout the installation, you're prompted to answer some questions. In each case, the default choice displays in square brackets and you need only press Enter to accept the default. If there are alternative responses, these are shown in round brackets; to choose one of these, type the response and press Enter.

For example:

```
Do you want to continue? (y/n) [n]:
```

The possible answers to this question are y or n. The default answer when you type nothing and press Enter is n.

Running the installer

If you are considering running the installation as a non-root user, we suggest you review this carefully as you will have to get a root user to manually complete some parts of the installation afterwards. We recommend installing as the root user. (If you're concerned about the changes that will be made to your system, refer to Changes made to your system.)

To start the installation, run:

```
./install
```

You need to:

- Confirm your acceptance of the license agreement by typing "yes" or "no". For more information about the license agreement, refer to License to use.
- Supply the location where the software is to be installed.

We recommend accepting the default installation path.

For more information, refer to Where to install.

Locating or installing unixODBC

We strongly recommend you use the unixODBC Driver Manager because:

- The installation script is designed to work with unixODBC and can automatically add Easysoft ODBC-Access Driver and data sources during the installation.

- Most applications and interfaces that support ODBC are compatible with unixODBC. The Easysoft ODBC-Access Driver and any data sources that you add during the installation are automatically available to your applications and interfaces therefore.
- The unixODBC project is currently led by Easysoft developer Nick Gorham. This means that there is a great deal of experience at Easysoft of unixODBC in general and of supporting the Easysoft ODBC-Access Driver running under unixODBC. It also means that if you find a problem in unixODBC, it's much easier for us to facilitate a fix.

The installation starts by searching for unixODBC. There are two possible outcomes here:

1. If the installation script finds unixODBC, the following message displays:

```
Found unixODBC under path and it is version n.n.n
```

2. If the installation script can't find unixODBC in the standard places, you will be asked whether you have it installed.

If unixODBC is installed, you need to provide the unixODBC installation path. Usually, the path required is the directory above where odbcinst is installed. For example, if odbcinst is in /opt/unixODBC/bin/odbcinst, the required path is /opt/unixODBC.

If unixODBC is not installed, you should install the unixODBC included with this distribution.

If you already have unixODBC installed, you do not have to install the unixODBC included with the distribution, but you might consider doing so if your version is older than the one we provide.

The unixODBC in the Easysoft ODBC-Access Driver distribution is not built with the default options in unixODBC's configure line.

Option	Description
--prefix=/etc	This means the default SYSTEM odbc.ini file where SYSTEM data sources are located is /etc/odbc.ini.
--enable-drivers=no	This means other ODBC drivers that come with unixODBC are not installed.
--enable-iconv=no	This means unixODBC does not look for libiconv. Warnings about not finding an iconv library were confusing our customers.
--enable-stats=no	Turns off unixODBC statistics, which use system semaphores to keep track of used handles. Many systems do not have sufficient semaphore resources to keep track of used handles.
--enable-readline=no	This turns off readline support in isql. We did this because it ties isql to the version of libreadline on the system we build on. We build on as old a version of the operating system as we can for forward compatibility. Many newer Linux systems no longer include the older readline libraries and so turning on readline support makes isql unusable on these systems.
--prefix=/usr/local/easysoft/unixODBC	This installs unixODBC into /usr/local/easysoft/unixODBC.

Installing the Easysoft ODBC driver

The Easysoft ODBC-Access Driver installation script:

- Installs the driver.
- Registers the driver with the unixODBC Driver Manager.

  If the Easysoft ODBC-Access Driver is already registered with unixODBC, a warning displays that lists the drivers unixODBC knows about. If you're installing the Easysoft ODBC-Access Driver into a different directory than it was installed before, you need to edit your odbcinst.ini file after the installation and correct the Driver and Setup paths. unixODBC's odbcinst doesn't update these paths if a driver is already registered.
- Creates an example Easysoft ODBC-Access Driver data source. If unixODBC is installed and you registered the Easysoft ODBC-Access Driver with unixODBC, the installation script adds example data source to your odbc.ini file.

Licensing

The *installation_path*/easysoft/license/licshell program lets you obtain or list licenses.

Licenses are stored in *installation_path*/easysoft/license/licenses.

> **Important** After obtaining a license, you should make a backup copy of this file.

The installation script asks you if you want to request an Easysoft ODBC-Access Driver license:

```
Would you like to request a Easysoft ODBC-Access Driver license now (y/n) [y]:
```

You do not need to obtain a license during the installation, you can run licshell after the installation to obtain or view licenses.

If you answer y, the installation runs the licshell script.

To obtain a license automatically, you need to be connected to the Internet and allow outgoing connections to license.easysoft.com on port 8884. If you're not connected to the Internet or don't allow outgoing connections on port 8884, the License Client can create a license request file that you can email to us.

When you start the License Client, the following menu displays:

```
[0] exit
[1] view existing license
[n] obtain a license for the desired product.
```

To obtain a license, select one of the options from [2] onwards for the product you're installing. The License Client then runs a program that generates a key that's used to identify the product and operating system (we need this key to license you).

After you have chosen the product to license (Easysoft ODBC-Access Driver), you need to supply:

- Your full name.
- Your company name.
- An email contact address. This must be the email address that you used when you registered on the Easysoft web site.
- A reference number (also referred to as an authorization code). When applying for a trial license, press Enter when prompted for a reference number. This field only applies to full (paid) licenses.

You're then asked to choose how you want to obtain the license.

The choices are:

- [1] Automatically by contacting the Easysoft License Daemon

  This requires a connection to the Internet and the ability to support an outgoing TCP/IP connection to license.easysoft.com on port 8884.

- [2] Write information to file

  The license request is output to license_request.txt.

- [3] Cancel this operation

If you choose to obtain the license automatically, the License Client tries to open a TCP/IP connection to license.easysoft.com on port 8884 and send the details you supplied along with your machine number. No other data is sent. The data sent is transmitted as plain text, so if you want to avoid the possibility of this information being intercepted by someone else on the Internet, you should choose [2] and send the the request to us. The License daemon returns the license key, prints it to the screen and make it available to the installation script in the file licenses.out.

If you choose option [2], the license request is written to the file license_request.txt. You should then exit the License Client by choosing option [0] and complete the installation. After you have sent the license request to us, we'll return a license key. Add this to the end of the file *installation_path*/easysoft/license/licenses.

Post installation steps for non-root installations

If you installed the Easysoft ODBC-Access Driver as a non-root user (not recommended), there may be some additional steps you need to do manually:

1. If you attempt to install the Easysoft ODBC-Access Driver under the unixODBC Driver Manager and you do not have write permission to unixODBC's odbcinst.ini file, the driver can't be added.

   You can manually install the driver under unixODBC by adding an entry to the odbcinst.ini file. Run odbcinst -j to find out the location of the DRIVERS file then append the lines from drv_template file to odbcinst.ini. (drv_template is in the directory where the Easysoft distribution was untarred to.)

2. No example data sources can be added into unixODBC if you do not have write permission to the SYSTEM odbc.ini file. Run odbcinst -j to find out the location of the SYSTEM DATA SOURCES file then add your data sources to this file.
3. On systems where the dynamic linker has a configuration file defining the locations where it looks for shared objects (Linux and FreeBSD), you need to add:

   ```
   installation_path/easysoft/lib
   installation_path/easysoft/unixODBC/lib
   ```

   The latter entry is only required if you installed the unixODBC included with this distribution. Sometimes, after changing the dynamic linker configuration file, you need to run a program to update the dynamic linker cache. (For example, /sbin/ldconfig on Linux.)

4. If you didn't install the Easysoft ODBC-Access Driver in the default location, you need to do one of the following:
   - Link /usr/local/easysoft to the easysoft directory in your chosen installation path.

     For example, if you installed in /home/user, the installation creates /home/user/easysoft and you need to symbolically link /usr/local/easysoft to /home/user/easysoft:

     ```
     ln -s /home/user/easysoft /usr/local/easysoft
     ```

- Set and export the EASYSOFT_ROOT environment variable to *installation_path*/easysoft.

5. If your system doesn't have a dynamic linker configuration file, you need to add the paths listed in step 3 to whatever environment path the dynamic linker uses to locate shared objects. You may want to add these paths to a system file run whenever someone logs. For example, /etc/profile.

   The environment variable depends on the dynamic linker. Refer to your ld or ld.so man page. It is usually:

   ```
   LD_LIBRARY_PATH, LIBPATH, LD_RUN_PATH, or SHLIB_PATH.
   ```

Uninstalling on Linux or UNIX

There is no automated way to remove the Easysoft ODBC-Access Driver in this release. However, removal is quite simple. To do this:

1. Change directory to *installation_path*/easysoft and delete the product directory. *installation_path* is the Easysoft ODBC-Access Driver installation directory, by default /usr/local.
2. If you had to add this path to the dynamic linker search paths (for example, /etc/ld.so.conf on Linux), remove it. You may have to run a linker command such as /sbin/ldconfig to get the dynamic linker to reread its configuration file. Usually, this step can only be done by the root user.
3. If you were using unixODBC, the Easysoft ODBC-Access Driver entry needs to be removed from the odbcinst.ini file. To check whether the Easysoft ODBC-Access Driver is configured under unixODBC, use odbcinst -q -d. If the command output contains [Easysoft ODBC-ACCESS], uninstall the driver from unixODBC by using:

```
odbcinst -u -d -n Easysoft ODBC-ACCESS
```

If a reduced usage count message is displayed, repeat this command until odbcinst reports that the driver has been removed.

1. If you created any Easysoft ODBC-Access Driver data sources under unixODBC, you may want to delete these. To do this, first use odbcinst -j to locate USER and SYSTEM odbc.ini files. Then check those files for data sources that have the driver attribute set to Easysoft ODBC-ACCESS.
2. Remove the install.info for the Easysoft ODBC-Access Driver from the /usr/local/easysoft directory.

Connecting to Access

Applications that support ODBC interface with an ODBC Driver Manager, which is included with the operating system, and also the Easysoft ODBC driver distribution on some platforms. One of the jobs that the ODBC Driver Manager does is to manage ODBC data sources. A data source specifies which ODBC driver to load, which data store to connect to, and how to connect to it.

Before setting up a data source, you must have successfully installed the Easysoft ODBC-Access Driver.

Connecting from Linux or UNIX

Creating an ODBC data source

There are two ways to create a data source to your Access data:

- Create a SYSTEM data source, which is available to anyone who logs on to the computer where the Easysoft ODBC-Access Driver is installed.

  – Or –

- Create a USER data source, which is only available to the user who is currently logged on to the computer where the Easysoft ODBC-Access Driver is installed.

By default, the Easysoft ODBC-Access Driver installation creates a sample SYSTEM data source named ACCESS_SAMPLE. If you're using the unixODBC included in the Easysoft ODBC-Access Driver distribution, the SYSTEM odbc.ini file is in /etc.

If you built unixODBC yourself, or installed it from some other source, SYSTEM data sources are stored in the path specified with the configure option --sysconfdir=*directory*. If sysconfdir was not specified when unixODBC was configured and built, it defaults to /usr/local/etc.

If you accepted the default choices when installing the Access, USER data sources must be created and edited in $HOME/.odbc.ini.

Notes

- To display the directory where unixODBC stores SYSTEM and USER data sources, type odbcinst -j.
- By default, you must be logged in as root to edit a SYSTEM data source defined in /etc/odbc.ini.

You can either edit the sample data source or create new data sources.

Each section of the odbc.ini file starts with a data source name in square brackets [ ] followed by a number of attribute=value pairs.

The Driver attribute identifies the ODBC driver in the odbcinst.ini file to use for a data source. When the Easysoft ODBC-Access Driver is installed into unixODBC, it places a Easysoft ODBC-ACCESS entry into the odbcinst.ini file. You should always have Driver = Easysoft ODBC-ACCESS in your Easysoft ODBC-Access Driver data sources therefore.

To configure a Easysoft ODBC-Access Driver data source, in your odbc.ini file, you need to specify:

- The path to the Access database file (.mdb or .accdb).

For example:

```
[ACCESS_SAMPLE]
Driver = Easysoft ODBC-ACCESS
mdbfile = /home/myuser/ms-access/Northwind.mdb
```

The Easysoft ODBC-Access Driver must be able to find the following shared objects:

- libodbcinst.so

  By default, this is located in /usr/local/easysoft/unixODBC/lib/.

- libeslicshr.so

  By default, this is located in /usr/local/easysoft/lib/.

- libessupp.so By default, this is located in /usr/local/easysoft/lib/.

You may need to set and export LD_LIBRARY_PATH, SHLIB_PATH, or LIBPATH (depending on your operating system and run-time linker) to include the directories where libodbcinst.so, libeslicshr.so, and libessupp.so are located.

The isql query tool lets you test your Easysoft ODBC-Access Driver data sources. To test the Easysoft ODBC-Access Driver connection:

1. Change directory into /usr/local/easysoft/unixODBC/bin.
2. Enter ./isql -v *data_source*, where *data_source* is the name of the target data source.
3. At the prompt, enter an SQL query. For example:

```
SQL> SELECT * FROM Suppliers;
```

   –Or–

4. Enter help to return a list of tables:

```
SQL> help
```

Connection attributes

Setting on Linux and UNIX

To configure an Access data source, in your odbc.ini file, you need to specify:

- The path to the Access database file (.mdb or .accdb).

For example:

```
[ACCESS_SAMPLE]
Driver = Easysoft ODBC-ACCESS
mdbfile = /home/myuser/ms-access/Northwind.mdb
```

-Or-

```
[ACCESS_SAMPLE]
Driver = Easysoft ODBC-ACCESS
mdbfile = /home/myuser/ms-access/Northwind.accdb
```

If you're using the Easysoft ODBC-Access Driver to open an database file that Windows users may also be accessing, specify these settings:

- The SMB URL for the database file.
- The path to the SMB client library on the Easysoft ODBC-Access Driver machine.
- The user name and password of a user who can access the Windows or Samba share where the database file is located.

For example:

```
[ACCESS_SAMPLE_WINDOWS_SHARE]
Driver = Easysoft ODBC-ACCESS
mdbfile = /mnt/accounts/access/Northwind.mdb
smbpath = smb://mswin_machine/accounts/access/Northwind.mdb
smblib = /usr/lib/libsmbclient.so
smbuser = mywindows_user
smbauth = mywindows_password
```

-Or-

```
[ACCESS_SAMPLE_SAMBA_SHARE]
Driver = Easysoft ODBC-ACCESS
mdbfile = /home/samba/sales/access/Northwind.mdb
smbpath = smb://samba_server/sales/access/Northwind.mdb
smblib = /usr/lib/libsmbclient.so
smbuser = my_samba_share_valid_user
smbauth = my_samba_share_valid_password
```

For more information about the Easysoft ODBC-Access Driver's data source attributes, refer to the following table.

Name	Value
Description	Some applications display this to help users identify a particular data source.

mdbfile = *value*	The path to the Access database file (.mdb or .accdb) on the computer where the Easysoft ODBC-Access Driver is installed.
	If the database file is stored on the same computer as the Easysoft ODBC-Access Driver, specify the path to the database file. For example:
	`# Path to .mdb file stored on`
	`# Easysoft ODBC-Access Driver computer.`
	`mdbfile = /home/myuser/ms_access/Northwind.mdb`
	`-Or-`
	`# Path to .accdb file stored on`
	`# Easysoft ODBC-Access Driver computer.`
	`mdbfile = /home/samba/ms_access/Northwind.accdb`
	If the database file is stored on a Windows computer, you need to share the directory where the file is located and then mount the Windows share from the Easysoft ODBC-Access Driver computer. Specify the mounted share's path with mdbfile, along with the remote database file name. For example:
	`# Local path to .mdb file stored on`
	`# Windows. /mnt/access_windows is local mount`
	`# point for a remote Windows share.`
	`mdbfile = /mnt/access_windows/Northwind.mdb.`

Name	Value
dbq = *value*	The path to the Access database file (.mdb or .accdb) on the computer where the Easysoft ODBC-Access Driver is installed. The dbq attribute is an alternative to the mdbfile attribute, and is provided for symmetry with Microsoft's Access ODBC driver. You don't need to specify both the dbq and mdbfile attributes in your data source.

smbpath = *value*	The SMB URL for the database file. This attribute (along with smblib, smbuser, and smbauth) is required when you open a database file that Windows users may also be accessing.
	
	Important When you specify a SMB URL, the Easysoft ODBC-Access Driver uses the libsmbclient library to: let Windows know that it has opened a database file; prevent Windows users from opening a database file it has opened for exclusive access. Without this mechanism, there is the potential for database file corruption when Windows users and Easysoft ODBC-Access Driver users are working simultaneously with same the database.
	
	Use this syntax for the SMB URL:
	
	smb://*host*/*share*/*path*/*filename*
	
	where:
	
	host is the name or IP address of the computer on which the share is located.
	
	share is the share name.
	
	path specifies any subdirectories under *share*.
	
	filename is the database file name, which should be the same one as specified by the mdbfile attribute.
	
	For example:
	
	`# SMB URL for a .mdb file shared by a Windows computer.`
	`smbpath = smb://mswin_box/accounts/access/Northwind.mdb`
	
	`# SMB URL for a .accdb file shared from a UNIX`
	`# computer through Samba server.`
	`smbpath = smb://unix_box/sales/access/Northwind.accdb`
	
	Note that SMB URLs for Samba shares are case sensitive; SMB URLs for Windows shares are not case sensitive.

Name	Value
smblib = *value*	The path to the libsmbclient library on the computer where the Easysoft ODBC-Access Driver is installed. For example: smblib = /usr/lib/libsmbclient.so libsmbclient is part of the Samba suite. The Easysoft ODBC-Access Driver uses libsmbclient to: • Allow Linux and UNIX users to safely edit a database file that Windows users may also be accessing. • Prevent Windows users from opening a database file it has opened for exclusive access. • Prevent users from locking records in Windows that are already locked in the Easysoft driver and vice versa.
smbuser = *value*	The name of a user who can access the share where the database file is located. If the database file is stored on a Samba share, specify a Samba user. If the database file is stored on a Windows share, specify a Windows user. The user must have either read or read-write permissions for the share. If the user only has read access to the share, set the readonly data source attribute to yes.
smbauth = *value*	The password for smbuser.
lockfile = *value*	By default, when you open a database file for shared access, the Easysoft ODBC-Access Driver uses a locking information file named *dbfiledir*/*dbfilename*.ldb. For example, if you open a database named Nwind.mdb, the Easysoft ODBC-Access Driver expects a locking information file named Nwind.ldb in the same directory as Nwind.mdb. Use the lockfile attribute to specify an alternative .ldb file path if directory permissions prevent the Easysoft ODBC-Access Driver from manipulating a .ldb file in the default directory. Specify a directory where all users who connect to this data source have write access to. For example: lockfile = /tmp/Nwind.ldb Don't use the lockfile attribute if users may access the database file from Windows at your site. Windows applications access database files through the Jet or Access database engine, which expects the dbfiledir/dbfilename.ldb convention for lock files. There must only be one lock file per database file.

Name	Value
readonly = yes | no	To open the database file for read-only access, set readonly to yes. When opened for read-only access, you can view the Access database, but not change it. By default, readonly is set to no, which means the Easysoft ODBC-Access Driver opens the database for read-write access. If you open a database for read-write access without setting the smb* data source attributes, the Easysoft ODBC-Access Driver returns the warning: read write access without SMB channel can potentially allow corruption of the MDB file The Easysoft ODBC-Access Driver uses the smb* attributes to allow users to safely edit a database file that Windows users may also be accessing. If your database is shared with Windows users, configure the smb* attributes before opening the database for read-write access. Otherwise, set readonly to yes.
double_precision = *num*	The decimal precision to return for a DOUBLE column that's bound as a char. For example: double_precision = 15 Set this attribute if DOUBLE column data returned by the Easysoft ODBC-Access Driver does not have the expected number of decimal places.

Name	Value
exclusive = yes | no	To open the database file for exclusive access, set exclusive to yes. By default, exclusive is set to no. **Notes for libsmbclient users** When you set exclusive to yes, the Easysoft ODBC-Access Driver uses libsmbclient to the mark the database as share mode DENY_ALL. This share mode denies all open requests on the database file, which prevents other users from opening the database while you are connected to the Easysoft ODBC-Access Driver data source. **Note** Using share mode DENY_ALL is the only way to prevent Windows users from opening a database that you have opened for exclusive access with the Easysoft ODBC-Access Driver. To enable the Easysoft ODBC-Access Driver to use libsmbclient, you need to set the smb* attributes in the data source. If you set exclusive to yes without setting the smb* attributes, the Easysoft ODBC-Access Driver cannot prevent other users from opening the database and so returns the warning: exclusive access without SMB channel can not exclude other access, and can potentially allow corruption of the MDB file **Notes for non-libsmbclient users** Setting exclusive to yes prevents other Easysoft ODBC-Access Driver users from opening the database file while you're connected to the Easysoft ODBC-Access Driver data source.
ignore_rel = yes | no	Whether the Easysoft ODBC-Access Driver takes into account any relationships defined for a particular table when doing inserts, updates, or deletes. For example, with ignore_rel set to yes, the Easysoft ODBC-Access Driver would prevent a record from being deleted if one of its columns was a foreign key for another table. By default, ignore_rel is set to no.
work_mem_size = *value*	The size of the memory cache, in megabytes (MB), where the Easysoft ODBC-Access Driver will store rows before storing them on disk. For example: work_mem_size = 10
work_dir_path = *value*	The directory where the Easysoft ODBC-Access Driver temporarily stores result set rows when work_mem_size is exceeded. For example: work_dir_path = /tmp

Name	Value

htime_pattern = *value*	Use htime_pattern to specify the format used for hash (#) quoted date constants in queries. Available formats are:
	`MDY (month/day/year)`
	`DMY (day/month/year)`
	`YMD (year/month/day)`
	For example:
	htime_pattern = MDY

unicode_map = 0 | 1 | 2	How Easysoft ODBC-Access Driver maps 16-bit characters to 8-bit characters. The unicode_map attribute affects: TEXT, MEMO, and HYPERLINK data in result sets; metadata (column names); DELETE, INSERT INTO, UPDATE statements (TEXT columns only); SQL statement parameters. Use unicode_map if you experience data loss or corruption when working with character data. The available values for the unicode_map attribute are: 0 When retrieving data, the Easysoft ODBC-Access Driver preserves the low 8 bits of the encoded character and discards the high 8 bits. When unicode_map is set to 0, some applications may replace non-ASCII characters retrieved from the Access database with question marks (?). 1 When retrieving data, the Easysoft ODBC-Access Driver preserves ASCII characters and replaces non-ASCII characters with question marks. For example, Forêts d'érables becomes For?ts d'?rables. 2 The Easysoft ODBC-Access Driver converts characters to UTF-8 when retrieving data and from UTF-8 when submitting data (for example, inserting data into TEXT columns). By default, unicode_map is set to 0.

Connection examples

In this section:

- Connecting to an Access database on Windows from Linux
- Connecting to an Access database located on a Samba share

Connecting to an Access database on Windows from Linux

1. On the Easysoft ODBC-Access Driver machine, use smbmount to mount the Windows share where the database file is located.

   For example:

```
# smbmount //mywindowsmachine/myshare /mylinuxmountpoint -o
username=mywindowsuser,rw,directio
```

Important	For read-write access to the database, you must specify the directio option when mounting the share. Check the man page for your version of smbmount to find out whether it supports the directio option. Specifying directio turns off

	inode data caching on files opened on the mount. The Easysoft ODBC-Access Driver relies on this functionality when managing concurrent access to the database file.

When mounting the share, you need to supply the user name and password of a Windows user who has read, write, create, and delete privileges for the share. When testing the Easysoft ODBC-Access Driver with a database file located in a Windows share, this requirement equated to these Windows permissions:

Name	Value
Share permissions	Shared folder permissions
Change Read	Modify Read Write

These permissions allow the Easysoft ODBC-Access Driver to write to or create a locking information file (.ldb).

2. Configure the Easysoft ODBC-Access Driver data source.

   Example data sources:

```
# This Easysoft ODBC-Access Driver data source opens an Access
# database that is stored on a Windows share. The data source
# opens the database for shared read/write access; other users can
# open the database therefore.
[ACCESS_SAMPLE_WINDOWS_SHARE]
Driver = Easysoft ODBC-ACCESS

# The Windows share (accounts) has been attached under /mnt on the
# Easysoft ODBC-Access Driver machine.
mdbfile = /mnt/accounts/access/Northwind.mdb

# The smb* attributes allow the Easysoft ODBC-Access Driver to
# manage concurrent access to the database.
# The SMB URL for the database on the Windows share.
smbpath = smb://mswin_machine/accounts/access/Northwind.mdb

# The path to the SMB client library on the
# Easysoft ODBC-Access Driver machine
smblib = /usr/lib/libsmbclient.so

# The user name and password of a Windows user who has read and
# write privileges to the share.
smbuser = mywindows_user
smbauth = mywindows_password

# Opens the database for shared read/write access.
readonly = no
exclusive = no
```

   -Or-

```
# This data source opens the same Access database for exclusive
```

```
# access; the Easysoft ODBC-Access Driver will prevent other
# users from accessing the database, therefore.
[ACCESS_SAMPLE_WINDOWS_SHARE]
Driver = Easysoft ODBC-ACCESS
mdbfile = /mnt/accounts/access/Northwind.mdb
smbpath = smb://mswin_machine/accounts/access/Northwind.mdb
smblib = /usr/lib/libsmbclient.so

# The user name and password of a Windows user who can access the
# share and who has read and write privileges to the .mdb file.
smbuser = mywindows_user
smbauth = mywindows_password

# Opens the database for exclusive read/write access.
readonly = no
exclusive = yes
```

-Or-

```
# This data source opens the same Access database for read-only
# access; the Easysoft ODBC-Access Driver will prevent its users
# from making updates to the database, therefore.
[ACCESS_SAMPLE_WINDOWS_SHARE]
Driver = Easysoft ODBC-ACCESS
mdbfile = /mnt/accounts/access/Northwind.mdb
smbpath = smb://mswin_machine/accounts/access/Northwind.mdb
smblib = /usr/lib/libsmbclient.so

# The user name and password of a Windows user who has read
# privilege to the share.
smbuser = mywindows_user
smbauth = mywindows_password

# Opens the database for read-only access.
readonly = yes
exclusive = no
```

For more information about Easysoft ODBC-Access Driver data source attributes, refer to Connection attributes.

Connecting to an Access database located on a Samba share

1. Copy the Access database file to a Samba share on the Easysoft ODBC-Access Driver machine.

   The Easysoft ODBC-Access Driver requires the following Samba configuration options to be set for the share in which the database file is located.

Option	Value
oplocks	yes This is the default setting.

Option	Value
posix locking	yes This is the default setting.
blocking locks	yes This is the default setting.
veto oplock files	/**.mdb/**.MDB/**.ldb/**.LDB/**.accdb/**.ACCDB

2. Configure the Easysoft ODBC-Access Driver data source.

   Example data sources:

```
# This Easysoft ODBC-Access Driver data source opens an Access
# database that is stored on a Samba share. The data source
# opens the database for shared read/write access; other users can
# open the database therefore.
[ACCESS_SAMPLE_SAMBA_SHARE]
Driver = Easysoft ODBC-ACCESS
mdbfile = /home/samba/sales/access/Northwind.mdb

# The smb* attributes allow the Easysoft ODBC-Access Driver to
# manage concurrent access to the database.
# The SMB URL for the database on the Samba share.
smbpath = smb://samba_server/sales/access/Northwind.mdb

# The path to the SMB client library on the
# Easysoft ODBC-Access Driver machine
smblib = /usr/lib/libsmbclient.so

# The user name and password of a Samba user who has read/write
# access to the share.
smbuser = my_samba_share_valid_user
smbauth = my_samba_share_valid_password

# Opens the database for shared read/write access.
readonly = no
exclusive = no
```

   -Or-

```
# This data source opens the same Access database for exclusive
# access; the Easysoft ODBC-Access Driver will prevent other
# users from accessing the database, therefore.
[ACCESS_SAMPLE_SAMBA_SHARE]
Driver = Easysoft ODBC-ACCESS
mdbfile = /home/samba/sales/access/Northwind.mdb
smbpath = smb://samba_server/sales/access/Northwind.mdb
smblib = /usr/lib/libsmbclient.so

# The user name and password of a Samba user who has read/write
# access to the share.
```

```
smbuser = my_samba_share_valid_user
smbauth = my_samba_share_valid_password

# Opens the database for exclusive read/write access.
readonly = no
exclusive = yes
```

-Or-

```
# This data source opens the same Access database for read-only
# access; the Easysoft ODBC-Access Driver will prevent its users
# from making updates to the database, therefore.
[ACCESS_SAMPLE_SAMBA_SHARE]
Driver = Easysoft ODBC-ACCESS
mdbfile = /home/samba/sales/access/Northwind.mdb
smbpath = smb://samba_server/sales/access/Northwind.mdb
smblib = /usr/lib/libsmbclient.so

# The user name and password of a Samba user who has read
# access to the share.
smbuser = my_samba_share_valid_user
smbauth = my_samba_share_valid_password

# Opens the database for read-only access.
readonly = yes
exclusive = no
```

Troubleshooting database connection problems

This section lists some common connection problems and their solutions.

- Failed to open MDB file
- Failed to open SMB channel
- Could not open/create lock file, check sharing permissions

Failed to open MDB file

The Easysoft ODBC-Access Driver can open either a local or remote Access database file (.mdb or .accdb). The path to the database file is specified in the Easysoft ODBC-Access Driver data source in /etc/odbc.ini.

In the Easysoft ODBC-Access Driver data source, check the mdbfile attribute value.

Use ls to check that the mdbfile attribute in your data source specifies a valid path and that the directory is accessible.

If the ls command output contains "permission denied", contact your administrator. The directory containing the database file needs to be accessible (execute permission set) to you.

If the path is not valid, check with your administrator whether the database file is located on another machine. (The Easysoft ODBC-Access Driver can open a database file that is located on a Windows share.) If the database file is remote, use the mount command to check whether the share has been attached on the Easysoft ODBC-Access Driver machine. There should be an entry in the mount command output that corresponds with the directory specified with mdbfile. For example:

```
$ mount
/dev/sda3 on / type ext3 (rw,errors=remount-ro)
proc on /proc type proc (rw,noexec,nosuid,nodev)
/sys on /sys type sysfs (rw,noexec,nosuid,nodev)
.
.
.
//mywindowsmachine/myshare on /mnt/myshare type smbfs (rw)

$ grep mdbfile /etc/odbc.ini
mdbfile = /mnt/myshare/Northwind.mdb
```

If the mount command does not show the mount point specified in the mdbfile attribute, consult with your administrator. The share on which the database file is located needs to be mounted.

If the database file is on a Windows share, the user specified when mounting the share needs to have read, write, create, and delete permissions for the share. When testing the Easysoft ODBC-Access Driver with a database file located in a Windows share, this requirement equated to these Windows permissions:

Share permissions	Shared folder permissions
Change Read	Modify Read Write

If the database file is located on UNIX, the Easysoft ODBC-Access Driver will also return a "Failed to open MDB file" if:

1. The user running the application that is connecting to the data source only has read access to a local database file.

   -Or-

   The user specified by smbuser only has read access to a database file located on a Samba share.

2. The readonly attribute in the data source is set to yes. (Change this attribute's value to no and try again.)

Failed to open SMB channel

The Easysoft ODBC-Access Driver uses the libsmbclient library, which is part of the Samba suite, to:

- Let Windows know that it has opened a database file (.mdb or .accdb).
- Prevent Windows users from opening a database file it has opened for exclusive access.

Without this mechanism, there is the potential for database file corruption when Windows users and Easysoft ODBC-Access Driver users are working with the database specified by mdbfile.

To configure the driver for use with libsmbclient you use the smb* attributes in the driver data source.

If the "Failed to open SMB channel" error contains "file not found", check the smbpath attribute in the data source. The smbpath attribute needs to specify the SMB URL for the database file specified with mdbfile. The SMB URL has the format smb://*host*/*share*/*path*/*filename*.

In the following example, a remote Access database named Northwind.mdb is located on the Easysoft ODBC-Access Driver machine in /mnt/myshare_mountdir. The mount command shows that /mnt/myshare_mountdir is the mount point for a share named myshare on a machine named myremotemachine.

```
$ grep mdbfile /etc/odbc.ini
mdbfile = /mnt/myshare_mountdir/Northwind.mdb

$ mount
//myremotemachine/myshare on /mnt/myshare_mountdir type smbfs (rw)
```

The SMB URL for the database file would therefore be:

```
smb://myremotemachine/myshare/Northwind.mdb
```

If the "Failed to open SMB channel" error contains "errno=13", check the smbuser and smbauth attributes in the data source.

If the database file is located on a Samba share, the smbuser and smbauth attributes need to specify the user name and password for a Samba user (created with Samba tools such as smbpasswd) who has read access to the share. To open a database file for read-write access (readonly data source attribute set to no), the Samba user also needs write access to the share).

If the database file is located in a Windows share, the smbuser and smbauth attributes need to specify the user name and password for a Windows user who can access the share. If smbuser only has read privilege to the database file (as opposed to read and write), the readonly attribute in the data source needs to be set to yes. Otherwise, an "errno=13" error will be returned.

Could not open/create lock file, check sharing permissions

To allow its users to safely edit a database file that Windows users may also be accessing and updating, the Easysoft ODBC-Access Driver uses a locking information file (.ldb).

The driver uses a .ldb file to:

- Determine which records in the database are locked and by whom.
- Let other applications know which records in the database are locked by a Easysoft ODBC-Access Driver user.

The Easysoft ODBC-Access Driver may have to create and delete a .ldb file as well as write to it. If directory permissions prevent the Easysoft ODBC-Access Driver from writing to or creating a .ldb file, the Easysoft ODBC-Access Driver may return a "Could not open/create lock file, check sharing permissions" error.

If you get this error when opening a database file that Windows users do not require access to, specify an alternative .ldb file directory with the lockfile data source attribute. Specify a directory where you and other users who connect to the data source have write access to. For example:

```
lockfile = /tmp/Nwind.ldb
```

Permissions summary

This table lists the various permissions that may need to be set to allow the Easysoft ODBC-Access Driver to open your Access database file (.mdb or .accdb).

Permission	Applies to	Type	Notes
execute write	database file directory	Local UNIX permissions on Easysoft ODBC-Access Driver machine	The local directory from which the database file is opened must be accessible (execute permission set) by the user who is connecting to the data source. The local directory must also be writeable by the user who is connecting to the data source if the database file is opened for shared access (exclusive data source attribute set to no).
read write	database file	Local UNIX permissions on Easysoft ODBC-Access Driver machine	The database file must be readable (read permission set) by the user who is connecting to the data source. If the database file is opened for read/write access (readonly data source attribute set to no), the database file must also be writeable (write permission set) by this user.

Permission	Applies to	Type	Notes
read write	Samba share containing database file	Samba	These permissions are only applicable if the database file is located on a Samba share on UNIX. The Samba user specified by the smbuser data source attribute needs to have read access to the share. If the database file is opened for read/write access (readonly data source attribute set to no), this user must also have write access to the share.
read write create delete	Windows share containing database file	Windows	These permissions are only applicable if the database file is located on a Windows share. The Windows user whose authentication details are supplied when mounting the share needs to have read, write, create, and delete privileges on the share. Note that on some versions of Windows, the create and delete privileges may be replaced by the modify privilege.

Permission	Applies to	Type	Notes
read write	Windows share containing database file	Windows	These permissions are only applicable if the database file is located on a Windows share. The Windows user specified by the smbuser data source attribute needs to have read privilege on the share. If the database file is opened for read/write access (readonly data source attribute set to no), this user must also have write privilege.

DSN-less connections

Some applications allow you to make an ODBC connection without configuring a data source. To do this, you supply a connection string that contains the ODBC driver name and other driver-specific attribute-value pairs.

Here's an example Easysoft ODBC-Access Driver connection string:

```
Driver=Easysoft ODBC-ACCESS;MDBFILE=/home/myuser/ms_access/Northwind.mdb;
```

For a list of the other attributes you can set in the connection string, refer to this section.

Logging

If you report an issue to us, we may ask you to turn on ODBC Driver Manager or Easysoft ODBC-Access Driver logging, to help us diagnose the cause of the issue.

To turn on logging, refer to the following sections.

Note	If your application is a service (for example, Oracle or SQL Server), you may need to restart the service before enabling logging takes effect. To do this on Linux or UNIX, use service, systemctl, or a vendor-supplied script. To do this on Windows, use the Windows **Services** app.

ODBC Driver Manager logging on Linux or UNIX

For the unixODBC Driver Manager, add the following attributes to the [ODBC] section (create one if none exists) in odbcinst.ini.

```
Trace = Yes
TraceFile = /path/filename
```

For example:

```
[ODBC]
Trace = Yes
TraceFile = /tmp/sql.log
```

Ensure that the user who's running the application to log has write permission to TraceFile (and to the directory containing it), otherwise no logging information will be produced.

Easysoft ODBC-Access Driver logging on Linux and UNIX

Driver manager trace files show all the ODBC calls an application makes, including their arguments and return values. Easysoft ODBC-Access Driver logging is specific to the Easysoft driver and is of most use when making a support call.

To turn on Easysoft ODBC-Access Driver logging, edit your ODBC data source in odbc.ini. For example:

```
[ACCESS_SAMPLE]
.
.
Logging = Yes
LogFile = /tmp/easysoft-odbc-driver.log
```

The value shown in the example specifies a log file named /tmp/easysoft-odbc-driver.log. Ensure that the user who's running the application to log has write permission to the log file (and to the directory containing it), otherwise no logging information will be produced.

Client applications

How to work with Access data in some example applications and programming languages:

- Oracle
- LibreOffice
- Go
- Node.js
- Perl
- PHP
- Python
- R

Oracle

1. Install the Easysoft ODBC-Access Driver on same computer as Oracle.
2. Configure an ODBC data source.
3. Follow the instructions for your Oracle platform.

Connecting Access to Oracle on Linux and UNIX

1. Create a DG4ODBC init file on your Oracle machine. To do this, change to the $ORACLE_HOME\hs\admin directory. Create a copy of the file initdg4odbc.ora. Name the new file initAccess.ora.

   > **Note** In these instructions, replace $ORACLE_HOME with the location of your Oracle HOME directory. For example, /u01/app/oracle/product/21c/dbhome_1.

2. Ensure these parameters and values are present in your init file:

   ```
   HS_FDS_CONNECT_INFO = "Access"
   ```

   Replace Access with the name of your Easysoft ODBC-Access Driver data source.

3. Comment out the line that enables DG4ODBC tracing. For example:

   ```
   #HS_FDS_TRACE_LEVEL = <trace_level>
   ```

4. Add an entry to $ORACLE_HOME/network/admin/listener.ora that creates a SID_NAME for DG4ODBC. For example:

   ```
   SID_LIST_LISTENER =
    (SID_LIST =
      (SID_DESC=
        (SID_NAME=Access)
        (ORACLE_HOME=$ORACLE_HOME)
        (PROGRAM=dg4odbc)
        (ENVS=LD_LIBRARY_PATH = /usr/local/easysoft/unixODBC/lib:
               /usr/local/easysoft/lib)
      )
    )
   ```

   Replace oracle_home_directory with the value of $ORACLE_HOME. For example, /u01/app/oracle/product/21c/dbhome_1.

5. Add a DG4ODBC entry to $ORACLE_HOME/network/admin/tnsnames.ora that specifies the SID_NAME created in the previous step. For example:

   ```
   Access =
     (DESCRIPTION =
       (ADDRESS = (PROTOCOL = TCP)(HOST = oracle_host)(PORT = 1521))
       (CONNECT_DATA =
         (SID = Access)
       )
       (HS = OK)
     )
   ```

   Replace oracle_host with the host name of your Oracle machine.

6. Start (or restart) the Oracle Listener:

```
cd $ORACLE_HOME/bin
./lsnrctl stop
./lsnrctl start
```

7. Connect to your Oracle database in SQL*Plus.
8. In SQL*Plus, create a database link for Access. For example:

```
CREATE PUBLIC DATABASE LINK AccessLink
    CONNECT TO "dbuser" IDENTIFIED BY "dbpassword"
    USING 'Access';
```

   Replace dbuser and dbpassword with your backend user name and password, if applicable.

9. Try querying and updating your Access data. For example:

```
SELECT "Surname" FROM "Customers"@AccessLink;

DECLARE
  num_rows integer;
BEGIN
num_rows:=DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE@AccessLink
('INSERT INTO Customers (Surname, GivenName, City, Phone, CompanyName) VALUES
(''Devlin'', ''Michaels'', ''Kingston'', ''2015558966'', ''PowerGroup'')');
END;
/

DECLARE
  num_rows integer;
BEGIN
num_rows:=DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE@AccessLink
('UPDATE "Customers" SET "Surname" = ''Jones'' WHERE "CompanyName" =
''PowerGroup''');
END;
/

DECLARE
  num_rows integer;
BEGIN
num_rows:=DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE@AccessLink
('DELETE from "Customers" WHERE CompanyName = ''PowerGroup''');
END;
/
```

Notes

- If you have problems connecting to Access from Oracle, enable DG4ODBC tracing and check the trace files written to the $ORACLE_HOME/hs/trace directory. To enable DG4ODBC tracing, add the line HS_FDS_TRACE_LEVEL = DEBUG to initAccess.ora and then start or restart the Oracle listener. If the trace directory does not exist, create it.

LibreOffice

1. Install the Easysoft ODBC-Access Driver on same computer as LibreOffice.
2. Configure an ODBC data source.
3. Choose **File > New > Database**.
4. Choose **Connect to an existing database**.
5. Choose **ODBC** in the list, and then choose **Next**.
6. Choose **Browse**, double-click your data source, and then choose **Next**.
7. If your database requires a database user name, enter it in the **User name** box. If this user needs to supply a password choose the **Password required** check box.
8. Choose **Finish**.
9. Save the database when prompted.

   The database opens in a new Base window. From here you can access your data.
10. In the left pane of the database window, choose the **Tables** icon to display a hierarchy of tables. Enter the database password if prompted, and then choose **OK**.
11. To retrieve the data in a table, in the **Tables** pane, double-click a table.
12. Choose the **Queries** icon to create a query.

    Use any of the methods listed in the **Tasks** pane to create a query.

Go

1. Install the Easysoft ODBC-Access Driver on same computer as Go.
2. Configure an ODBC data source.
3. Install the odbc package for Go:

```
go mod init test
go get github.com/alexbrainman/odbc
```

4. Create and then use Go to run this script, which retrieves some Access data:

```
package main

import (
    _ "github.com/alexbrainman/odbc"
    "database/sql"
        "log"
)

func main() {
        // Replace the DSN value with the name of your ODBC data source.
        db, err := sql.Open("odbc",
                "DSN=Access")
        if err != nil {
                log.Fatal(err)
        }

        var (
                name string
        )

        rows, err := db.Query("SELECT Surname FROM Customers")
        if err != nil {
                log.Fatal(err)
        }
        defer rows.Close()
        for rows.Next() {
                err := rows.Scan(&name)
                if err != nil {
                        log.Fatal(err)
                }
                log.Println(name)
        }
        err = rows.Err()
        if err != nil {
                log.Fatal(err)
        }

        defer db.Close()
}
```

Node.js

1. Install the Easysoft ODBC-Access Driver on same computer as Node.js.
2. Configure an ODBC data source.
3. Install the odbc module for Node.js:

```
npm install odbc
```

4. Create and then use Node.js to run this script, which retrieves some Access data:

```
const odbc = require('odbc');
// Replace Access with the name of your Easysoft ODBC-Access Driver
// data source.
const connection = odbc.connect('DSN=Access', (error, connection) => {
    connection.query('SELECT Surname FROM Customers', (error, result) => {
        if (error) { console.error(error) }
        console.log(result);
    });
});
```

5. This script retrieves the tables and views in your Easysoft ODBC-Access Driver data source:

```
const odbc = require('odbc');
const connection = odbc.connect('DSN=Access', (error, connection) => {
    connection.tables(null, null, null, null, (error, result) => {
        if (error) { return; }
        const util = require('util');
        console.log(util.inspect(result, {maxArrayLength: null, depth:null}))
    });
});
```

6. This script retrieves the names of the columns in these tables and views:

```
const odbc = require('odbc');
const connection = odbc.connect('DSN=Access', (error, connection) => {
    connection.columns(null, null, null, null, (error, result) => {
        if (error) { return; }
        const util = require('util');
        console.log(util.inspect(result, {maxArrayLength: null, depth:null}))
    });
});
```

7. These scripts insert, update, and then delete some Access data:

```
const odbc = require("odbc");
const connection = odbc.connect("DSN=Access", (error, connection) => {
    connection.query("INSERT INTO
    Customers (
        Surname,
        GivenName,
        City,
        Phone,
        CompanyName
```

```
    )
VALUES
    (
        'Devlin',
        'Michaels',
        'Kingston',
        '2015558966',
        'PowerGroup'
    )", (error, result) => {
        if (error) { console.error(error) }
        console.log(result);
    });
});

const odbc = require("odbc");
const connection = odbc.connect("DSN=Access", (error, connection) => {
    connection.query("UPDATE Customers SET Surname = 'Jones' WHERE CompanyName =
'PowerGroup'", (error, result) => {
        if (error) { console.error(error) }
        console.log(result);
    });
});

const odbc = require("odbc");
const connection = odbc.connect("DSN=Access", (error, connection) => {
    connection.query("DELETE FROM Customers WHERE CompanyName = 'PowerGroup'",
(error, result) => {
        if (error) { console.error(error) }
        console.log(result);
    });
});
```

Perl

1. Install the Easysoft ODBC-Access Driver on same computer as Perl.
2. Configure an ODBC data source.
3. Check whether your Perl distribution supports ODBC:

```
perl -e 'use DBD::ODBC;'
```

4. Do one of the following:
   - If you get no output, your Perl distribution supports ODBC. Skip to the next step.
   - If you get:

     ```
     Can't locate DBD/ODBC.pm
     ```

     you need to install DBD::ODBC before you can use the Easysoft ODBC-Access Driver to connect to Access.
5. Create and then use Perl to run this script, which retrieves some Access data:

```
use strict;
use DBI;
# Replace Access with the name of your Easysoft ODBC-Access Driver data source.
my $dbh = DBI-> connect('dbi:ODBC:Access');

my $sql = "SELECT Surname FROM Customers";

my $sth = $dbh->prepare($sql)
    or die "Can't prepare statement: $DBI::errstr";

$sth->execute();

my($Col);

# Fetch and display the result set values.
while(($Col) = $sth->fetchrow()){
   print("$Col\n");
}

$dbh->disconnect if ($dbh);
```

6. This script retrieves the tables and views in your Easysoft ODBC-Access Driver data source:

```
use strict;
use DBI;
my $dbh = DBI-> connect('dbi:ODBC:Access');

my $sth = $dbh->table_info()
    or die "Can't prepare statement: $DBI::errstr";

my @row;

while (@row = $sth->fetchrow_array) {
    print join(", ", @row), "\n";
```

```
}

$dbh->disconnect if ($dbh);
```

7. This script retrieves the names of the columns in these tables and views:

```
use strict;
use DBI;
my $dbh = DBI-> connect('dbi:ODBC:Access');

my $sth = $dbh->column_info('','','','')
    or die "Can't prepare statement: $DBI::errstr";

my @row;
while (@row = $sth->fetchrow_array) {
    print join(", ", @row), "\n";
}

$dbh->disconnect if ($dbh);
```

8. These scripts insert, update, and then delete some Access data:

```
use strict;
use DBI;
my $dbh = DBI-> connect('dbi:ODBC:Access');

my $sth = $dbh->prepare(q/INSERT INTO Customers (Surname, GivenName, City, Phone,
CompanyName) VALUES (?, ?, ?, ?, ?)/)
    or die "Can't prepare statement: $DBI::errstr";

$sth->execute('Devlin', 'Michaels', 'Kingston', '2015558966', 'PowerGroup');

$dbh->disconnect if ($dbh);

use strict;
use DBI;
my $dbh = DBI-> connect('dbi:ODBC:Access');

my $sth = $dbh->prepare('UPDATE Customers SET Surname = \'Jones\' WHERE
CompanyName = ?')
    or die "Can't prepare statement: $DBI::errstr";

$sth->execute('PowerGroup');

$dbh->disconnect if ($dbh);

use strict;
use DBI;
my $dbh = DBI-> connect('dbi:ODBC:Access');

my $sth = $dbh->prepare('DELETE FROM Customers WHERE CompanyName = ?')
    or die "Can't prepare statement: $DBI::errstr";
```

```
$sth->execute('PowerGroup');

$dbh->disconnect if ($dbh);
```

Further information

- Perl DBI DBD::ODBC tutorial: Drivers, data sources, and connection

PHP

1. Install the Easysoft ODBC-Access Driver on same computer as PHP.
2. Configure an ODBC data source.
3. Check whether your PHP distribution supports ODBC. In php.ini, make sure there is no comment character (;) before the extension_dir and extension=odbc settings (;extension_dir=*directory* becomes extension_dir=*directory* and ;extension=odbc becomes extension=odbc).
4. Create and then use PHP to run this script, which retrieves some Access data:

```
<?php
  // Replace Access with the name of your Easysoft ODBC-Access Driver data
source.
  //  If your database requires a user name and password, supply them in the
odbc_connect_call.
  $con = odbc_connect("Access", "", "");
  $stmt = odbc_exec($con, "SELECT * FROM Customers");
  // You may need to change the capitalisation of Surname to all upper case or
all lower case.
  while ($row = odbc_fetch_array($stmt)) {
      echo "Surname = ", $row["Surname"], "\n";
  }
  odbc_close($con);
?>
```

5. This script retrieves the tables and views in your Easysoft ODBC-Access Driver data source:

```
<?php
$con = odbc_connect("Access", "", "");
$tables = odbc_tables($con);
while (($row = odbc_fetch_array($tables))) {
    print_r($row);
}
odbc_close($con);
?>
```

6. This script retrieves the names of the columns in these tables and views:

```
<?php
$con = odbc_connect("Access", "", "");
$columns = odbc_columns($con);
while (($row = odbc_fetch_array($columns))) {
    print_r($row);
}
odbc_close($con);
?>
```

7. These scripts insert, update, and then delete some Access data:

```
<?php
  $cnx = odbc_connect("Access", "", "");
  $stmt = odbc_prepare($cnx, "INSERT INTO Customers (Surname, GivenName, City,
Phone, CompanyName) VALUES (?, ?, ?, ?, ?)");
```

```
  $success = odbc_execute($stmt, array('Devlin', 'Michaels', 'Kingston',
'2015558966', 'PowerGroup'));
  odbc_close($cnx);
?>

<?php
  $cnx = odbc_connect("Access", "", "");
  $stmt = odbc_prepare($cnx, "UPDATE Customers SET Surname = 'Jones' WHERE
CompanyName = ?");
  $success = odbc_execute($stmt, array('PowerGroup'));
  odbc_close($cnx);
?>

<?php
  $cnx = odbc_connect("Access", "", "");
  $stmt = odbc_prepare($cnx, "DELETE FROM Customers WHERE CompanyName = ?");
  $success = odbc_execute($stmt, array('PowerGroup'));
  odbc_close($cnx);
?>
```

Further information

- Easysoft PHP tutorials and code samples

Python

1. Install the Easysoft ODBC-Access Driver on same computer as Python.
2. Configure an ODBC data source.
3. Check whether your Python distribution supports ODBC.

```
pip list
```

If you don't have pip installed:

```
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
python get-pip.py
```

4. Do one of the following:
   - If the output contains pyodbc, your Python distribution supports ODBC. Skip to the next step.
   - If the output does not contain pyodbc, use pip to install this module:

```
pip install pyodbc
```

5. Create and then use Python to run this script, which retrieves some Access data:

```
import pyodbc

# Replace Access with the name of your Easysoft ODBC-Access Driver data source.
cnxn = pyodbc.connect("DSN=Access")
cursor = cnxn.cursor()
sql = "SELECT Surname FROM Customers"
cursor.execute(sql)
rows = cursor.fetchall()
# You may need to change the capitalisation of Surname to all upper case or all
lower case.
for row in rows:
     print(row.Surname)
exit()
```

6. This script retrieves the tables and views in your Easysoft ODBC-Access Driver data source:

```
import pyodbc

# Replace Access with the name of your Easysoft ODBC-Access Driver data source.
cnxn = pyodbc.connect("DSN=Access")
cursor = cnxn.cursor()
cursor.tables()
rows = cursor.fetchall()
for row in rows:
     print(row.table_name)
exit()
```

7. This script retrieves the names of the columns in these tables and views:

```
import pyodbc

# Replace Access with the name of your Easysoft ODBC-Access Driver data source.
```

```
cnxn = pyodbc.connect("DSN=Access")
cursor = cnxn.cursor()
cursor.columns()
rows = cursor.fetchall()
for row in rows:
     print(row.table_name, row.column_name)
exit()
```

8. These scripts insert, update, and then delete some Access data:

```
import pyodbc

cnxn = pyodbc.connect("DSN=Access")
cursor = cnxn.cursor()
sql = "INSERT INTO Customers (Surname, GivenName, City, Phone, CompanyName)
VALUES (?, ?, ?, ?, ?)"
cursor.execute(sql, 'Devlin', 'Michaels', 'Kingston', '2015558966', 'PowerGroup')
cursor.commit()
exit()
```

```
import pyodbc

cnxn = pyodbc.connect("DSN=Access")
cursor = cnxn.cursor()
sql = "UPDATE Customers SET Surname = 'Jones' WHERE CompanyName = ?"
cursor.execute(sql, 'PowerGroup')
cursor.commit()
exit()
```

```
import pyodbc

cnxn = pyodbc.connect("DSN=Access")
cursor = cnxn.cursor()
sql = "DELETE FROM Customers WHERE CompanyName = ?"
cursor.execute(sql, 'PowerGroup')
cursor.commit()
exit()
```

Further information

- Easysoft Python tutorials and code samples

R

1. Install the Easysoft ODBC-Access Driver on same computer as R.
2. Configure an ODBC data source.
3. In R Console, check whether your R distribution supports ODBC.

```
library("RODBC")
```

4. Do one of the following:
   - If you get no output, you have the ODBC library for R. Skip to the next step.
   - If you get an "there is no package" error, install the ODBC library for R:

```
install.packages("RODBC")
```

5. Create and then use R to run this script, which retrieves some Access data:

```
library("RODBC")
# Replace Access with the name of your Easysoft ODBC-Access Driver data source.
ch <- odbcConnect("Access")
sqlQuery(ch, paste("SELECT Surname FROM Customers"))
odbcClose(ch)
quit()
```

6. This script retrieves the tables and views in your Easysoft ODBC-Access Driver data source:

```
library("RODBC")
# Replace Access with the name of your Easysoft ODBC-Access Driver data source.
ch <- odbcConnect("Access")
sqlTables(ch)
odbcClose(ch)
quit()
```

7. This script retrieves the names of the columns in the specified table or view:

```
library("RODBC")
# Replace Access with the name of your Easysoft ODBC-Access Driver data source.
ch <- odbcConnect("Access")
# You may need to change the capitalisation of Customers to all upper case or all
lower case.
sqlColumns(ch, sqtable="Customers")
odbcClose(ch)
quit()
```

8. These scripts insert, update, and then delete some Access data:

```
library("RODBC")
ch <- odbcConnect("Access")
sqlQuery(ch, paste("INSERT INTO Customers (Surname, GivenName, City, Phone,
CompanyName) VALUES ('Devlin', 'Michaels', 'Kingston', '2015558966',
'PowerGroup')"))
odbcClose(ch)
quit()
```

```
library("RODBC")
ch <- odbcConnect("Access")
sqlQuery(ch, paste("UPDATE Customers SET Surname = 'Jones' WHERE CompanyName =
'PowerGroup'"))
odbcClose(ch)
quit()

library("RODBC")
ch <- odbcConnect("Access")
sqlQuery(ch, paste("DELETE FROM Customers WHERE CompanyName = 'PowerGroup'"))
odbcClose(ch)
quit()
```

About the Easysoft ODBC-Access Driver

The Easysoft ODBC-Access Driver provides real-time access to Access data from any application that supports ODBC.

- ODBC API and scalar functions
- Data type mapping
- SQL examples

ODBC API and scalar functions

API functions

Use this table to find out what ODBC API functions the Easysoft ODBC-Access Driver supports:

Function	Status
SQLAllocConnect	Supported
SQLAllocEnv	Supported
SQLAllocHandle	Supported
SQLAllocStmt	Supported
SQLBindCol	Supported
SQLBindParameter	Supported
SQLBrowseConnect	Not supported
SQLBulkOperations	Not supported
SQLCancel	Supported
SQLCloseCursor	Supported
SQLColAttribute	Supported
SQLColAttributes	Supported
SQLColumnPrivileges	Not supported
SQLColumns	Supported
SQLConnect	Supported
SQLCopyDesc	Supported
SQLDisconnect	Supported
SQLDriverConnect	Supported
SQLDrivers	Supported
SQLEndTran	Supported
SQLError	Supported
SQLExecDirect	Supported
SQLExecute	Supported
SQLExtendedFetch	Supported
SQLFetch	Supported
SQLFetchScroll	Supported
SQLForeignKeys	Supported
SQLFreeConnect	Supported
SQLFreeEnv	Supported
SQLFreeHandle	Supported
SQLFreeStmt	Supported

Function	Status
SQLGetConnectAtt	Supported
SQLGetConnectOption	Supported
SQLGetCursorName	Supported
SQLGetData	Supported
SQLGetDescField	Supported
SQLGetDescRec	Supported
SQLGetDiagField	Supported
SQLGetDiagRec	Supported
SQLGetEnvAttr	Supported
SQLGetFunctions	Supported
SQLGetInfo	Supported
SQLGetStmtAttr	Supported
SQLGetStmtOption	Supported
SQLGetTypeInfo	Supported
SQLMoreResults	Supported
SQLNativeSql	Supported
SQLNumParams	Supported
SQLNumResultCols	Supported
SQLParamData	Supported
SQLParamOptions	Supported
SQLPrepare	Supported
SQLPrimaryKeys	Supported
SQLProcedureColumns	Supported
SQLProcedures	Supported
SQLPutData	Supported
SQLRowCount	Supported
SQLSetConnectAttr	Supported
SQLSetConnectOption	Supported
SQLSetCursorName	Supported
SQLSetDescField	Supported
SQLSetDescRec	Supported
SQLSetEnvAttr	Supported
SQLSetParam	Supported
SQLSetPos	Supported
SQLSetScrollOptions	Supported

Function	Status
SQLSetStmtOption	Supported
SQLSetStmtAttr	Supported
SQLStatistics	Supported
SQLTablePrivileges	Not supported
SQLTables	Supported
SQLTransact	Supported

Scalar functions

The Easysoft ODBC-Access Driver supports a number of scalar functions:

- String functions
- Numeric functions
- Time, date, and interval functions
- System functions
- Conversion functions

Use either the SQL-92 syntax with scalar functions. For example:

```
SELECT
  Invoice_Id,
  Customer_Name,
  EXTRACT(YEAR FROM Due_Date) as "Year"
FROM
  Invoice
```

String functions

The Easysoft ODBC-Access Driver supports these string functions:

- ASCII(*string_exp*)
- BIT_LENGTH(*string_exp*)
- CHAR(*code*)
- CHAR_LENGTH(*string_exp*)
- CHARACTER_LENGTH(*string_exp*)`
- CONCAT(*string_exp1*, *string_exp2*)
- INSERT(*string_exp1*, *start*, *length*, *string_exp2*)
- LCASE(*string_exp*)
- LEFT(*string_exp*, *count*)
- LENGTH(*string_exp*)
- LOCATE(*string_exp1*, *string_exp2[,start]*)
- LTRIM(*string_exp*)
- OCTET_LENGTH(*string_exp*)
- POSITION(*char_exp IN char_exp*)
- REPEAT(*string_exp*, *count*)
- REPLACE(*string_exp1*, *string_exp2*, *string_exp3*)
- RIGHT(*string_exp*, *count*)
- RTRIM(*string_exp*)
- SOUNDEX(*string_exp*)
- SPACE(*count*)

- UCASE(*string_exp*)

Numeric functions

The Easysoft ODBC-Access Driver supports these numeric functions:

- ABS(*numeric_exp*)
- ACOS(*float_exp*)
- ASIN(*float_exp*)
- ATAN(*float_exp*)
- CEILING(*numeric_exp*)
- COS(*float_exp*)
- COT(*float_exp*)
- DEGREES(*numeric_exp*)
- EXP(*float_exp*)
- FLOOR(*numeric_exp*)
- LOG(*float_exp*)
- LOG10(*float_exp*)
- MOD(*integer_exp1*, *integer_exp2*)
- PI()
- POWER(*numeric_exp*, *integer_exp*)
- RADIANS(*numeric_exp*)
- RAND(*[integer_exp]*)
- ROUND(*numeric_exp*, *integer_exp*)
- SIGN(*numeric_exp*)
- SIN(*float_exp*)
- SQRT(*float_exp*)
- TAN(*float_exp*)
- TRUNCATE(*numeric_exp*, *integer_exp*)

Time, date, and interval functions

The Easysoft ODBC-Access Driver supports these time, date, and interval functions:

- CURRENT_DATE()
- CURRENT_TIME[(*time-precision*)]
- CURRENT_TIMESTAMP[(*timestamp-precision*)]
- DAYNAME(*date_exp*)
- DAYOFMONTH(*date_exp*)
- DAYOFWEEK(*date_exp*)
- DAYOFYEAR(*date_exp*)
- EXTRACT(*extract-field* FROM *extract-source*)
- HOUR(*time_exp*)
- MINUTE(*time_exp*)
- MONTH(*date_exp*)
- MONTHNAME(*date_exp*)
- QUARTER(*date_exp*)
- SECOND(*time_exp*)
- TIMESTAMPADD(*interval*, *integer_exp*, *timestamp_exp*)
- TIMESTAMPDIFF(*interval*, *timestamp_exp1*, *timestamp_exp2*)
- WEEK(*date_exp*)
- YEAR(*date_exp*)

System functions

The Easysoft ODBC-Access Driver supports these system functions:

- DATABASE()
- USER()

Conversion functions

The Easysoft ODBC-Access Driver supports supports the SQL-92 CAST function for conversion between compatible data types.

Data type mapping

The Easysoft ODBC-Access Driver maps Access data types to ODBC data types in this way:

Access data type	ODBC data type
BIT	SQL_Bit
BYTE	SQL_TINYINT
LONGBINARY	SQL_LONGVARBINARY
VARBINARY	SQL_VARBINARY
BINARY	SQL_BINARY
LONGCHAR	SQL_LONGVARCHAR
CHAR	SQL_CHAR
CURRENCY	SQL_NUMERIC
INTEGER	SQL_INTEGER
COUNTER	SQL_INTEGER
SMALLINT	SQL_SMALLINT
REAL	SQL_REAL
DOUBLE	SQL_DOUBLE
VARCHAR	SQL_VARCHAR
DATETIME	SQL_TYPE_TIMESTAMP`

SQL examples

Example queries

- To fetch all records from a table, use the asterisk symbol (*) in your queries. For example:

```
SELECT * FROM Customers
```

- To only fetch records whose values are different, use DISTINCT. For example:

```
-- Which different sales regions are there?
SELECT DISTINCT Region AS Different_Regions FROM SalesOrders
-- How many different sales regions are there?
SELECT COUNT(DISTINCT Region) AS Different_Regions FROM SalesOrders
```

- To filter records, use WHERE. For example:

```
SELECT
  OrderDate,
  SalesRepresentative
FROM
  SalesOrders
WHERE
  Region = 'Eastern'

SELECT
  OrderDate,
  SalesRepresentative
FROM
  SalesOrders
WHERE
  Region = 'Eastern'
  OR Region = 'Western'

SELECT
  OrderDate,
  SalesRepresentative
FROM
  SalesOrders
WHERE
  Region = 'Eastern'
  AND EXTRACT(YEAR FROM OrderDate) = 2025
```

You can also supply a WHERE clause value as a parameter. For example, to do this in Python:

```
cursor.execute("SELECT
  OrderDate,
  SalesRepresentative
FROM
  SalesOrders
WHERE
  Region = ?", ['Eastern'])
```

- To fetch records that don’t match the WHERE clause pattern use NOT. For example:

```
SELECT
  OrderDate,
  SalesRepresentative
FROM
  SalesOrders
WHERE
  NOT Region = 'Eastern'
```

- To sort the result set in either ascending or descending order, use ORDER BY. For example:

```
SELECT
  *
FROM
  SalesOrders
ORDER BY
  OrderDate ASC

SELECT
  *
FROM
  Contacts
ORDER BY
  (
    CASE
      WHEN Surname IS NULL THEN Title
      ELSE Surname
    END
  );
```

- To group a result set into summary rows, use GROUP BY. For example:

```
SELECT
    COUNT(Id) As "Number",
    ProductID
FROM
    SalesOrderItems
GROUP BY
    ProductID

SELECT
    COUNT(Id) As "Number",
    ProductID
FROM
    SalesOrderItems
GROUP BY
    ProductID
HAVING
  COUNT(Id) > 100;
```

- To do calculations based on result set vales, use the SQL aggregate functions MIN(), MAX(), COUNT(), SUM(), and AVG(). For example:

```
SELECT Max(Quantity) FROM SalesOrderItems
SELECT Sum(Quantity) FROM SalesOrderItems
```

- To convert between compatible data types, use CAST. For example:

```
SELECT CAST(Quantity AS Char(100))FROM  SalesOrderItems
```

- To fetch records that contain column values between a given range, use BETWEEN. For example:

```
SELECT ProductID FROM SalesOrderItems WHERE Quantity BETWEEN 10 AND 20
```

- To combine the result set of two or more SELECT statements, use UNION. For example:

```
SELECT City FROM Contacts
UNION
SELECT City FROM Customers
```

- To combine rows from two or more tables, use JOIN. For example:

```
SELECT SalesOrders.ID, Customers.Surname, SalesOrders.OrderDate
FROM SalesOrders
INNER JOIN Customers ON SalesOrders.CustomerID=Customers.ID;
```

- To fetch records that contain column values matching a search pattern, use LIKE. For example:

```
SELECT Surname, GivenName FROM Customers WHERE CompanyName LIKE 'R%'
SELECT Surname, GivenName FROM Customers WHERE CompanyName LIKE '_he'
```

- To search for columns without a value (NULL) or with a value (non NULL), use either IS NULL or IS NOT NULL. For example:

```
SELECT * FROM Customers WHERE CompanyName IS NULL
```

- To specify multiple values in a WHERE clause, you can use IN as an alternative to OR. For example:

```
SELECT
  OrderDate,
  SalesRepresentative
FROM
  SalesOrders
WHERE
  Region = 'Eastern'
  OR Region = 'Western'
  OR Region = 'Central'
```

  can be replaced with:

```
SELECT
  OrderDate,
  SalesRepresentative
FROM
  SalesOrders
WHERE
  Region IN  ('Eastern', 'Western', 'Central')
```

- To set the maximum number of records to return, use TOP. For example:

```
SELECT TOP 10 * FROM Customers
```

- To test for the existence of records in a subquery, use EXISTS. For example:

```
SELECT
  Name
FROM
  Products
WHERE
  EXISTS (
    SELECT
     *
    FROM
      SalesOrderItems
    WHERE
     Products.ID = SalesOrderItems.ProductID
      AND Quantity < 20
  )
```

Example inserts, updates, and deletes

- To insert a Access record, use INSERT INTO. For example:

```
INSERT INTO
    Customers (
        Surname,
        GivenName,
        City,
        Phone,
        CompanyName
    )
VALUES
    (
        'Devlin',
        'Michaels',
        'Kingston',
        '2015558966',
        'PowerGroup'
    )
```

- Here's an Oracle linked table example:

```
DECLARE
  num_rows integer;
BEGIN
num_rows:=DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE@AccessLink
('INSERT INTO Customers (Surname, GivenName, City, Phone, CompanyName) VALUES
(''Devlin'', ''Michaels'', ''Kingston'', ''2015558966'', ''PowerGroup'')');
END;
/
```

- The Easysoft ODBC-Access Driver also supports parameterized inserts. Here's an example of doing this in Perl:

```
my $sth = $dbh->prepare(q/INSERT INTO Customers (Surname, GivenName, City, Phone,
CompanyName) VALUES (?, ?, ?, ?, ?)/)
    or die "Can't prepare statement: $DBI::errstr";

$sth->execute('Devlin', 'Michaels', 'Kingston', '2015558966', 'PowerGroup');
```

- To update a Access record, use UPDATE. For example:

```
UPDATE Customers
SET
  Surname = 'Jones'
WHERE
  Account_Id = 'PowerGroup'
```

  The Easysoft ODBC-Access Driver also supports parameterized updates. Here's an example of doing this in Perl:

```
my $sth = $dbh->prepare('UPDATE Customers SET Surname = \'Jones\' WHERE
CompanyName = ?')
    or die "Can't prepare statement: $DBI::errstr";

$sth->execute('PowerGroup');
```

- To delete a Access record, use DELETE. For example:

```
-- Delete (mark inactive) a bank account
DELETE FROM Customers WHERE CompanyName = 'PowerGroup'
```

The Easysoft ODBC-Access Driver also supports parameterized deletes. Here's an example of doing this in Python:

```
sql = "DELETE FROM Customers WHERE CompanyName = ?"
cursor.execute(sql, 'PowerGroup')
```

Index

A
attribute, 20

B
Base
  working with Access data, 36

C
connecting to Access, 14
connection string attributes, 31

D
data source attributes, 15
data type mapping, 55
dbq attribute, 17
Description attribute, 16
DG4ODBC
  working with Access data, 34
double_precision attribute, 19
DSN-less connections, 31

E
Easysoft ODBC-Access Driver
  adding ODBC data sources
    on Linux or UNIX, 14
  connecting to Access, 14
  connection examples, 21
  data source attributes, 15
  data type mapping, 55
  DSN-less connections, 31
  installing
    on Linux or UNIX, 5
  logging, 32
  ODBC API support, 50
  scalar function support, 52
  SQL examples, 56
  troubleshooting, 25
  uninstalling
    on Linux or UNIX, 13
example connections, 21
exclusive attribute, 20

G
Go
  working with Access data, 37

H
htime_pattern attribute, 21

I
ignore_rel attribute, 20, 20
installing the Easysoft ODBC-Access Driver, 5

L
LibreOffice
  working with Access data, 36
lockfile attribute, 18
log files, 32

M
mdbfile attribute, 16

N
Node.js
  working with Access data, 38

O
ODBC API function support, 50
ODBC connection string attributes, 31
ODBC data sources
  adding
    on Linux or UNIX, 14
Oracle
  working with Access data, 34

P
Perl
  working with Access data, 40
PHP
  working with Access data, 43
Python
  working with Access data, 45

R
R
  working with Access data, 47
readonly attribute, 19

S
scalar function support, 52
smbauth attribute, 18
smbpath attribute, 17, 18
smbuser attribute, 18
SQL examples, 56

T
trace files, 32
troubleshooting the connection, 25

U
unicode_map attribute, 21
uninstalling
  on Linux or UNIX, 13