

Introduction to

easysoft[®] ODBC for RMS

version 1.5



Topics covered

Module 1 Introduction to ODBC (Open Database Connectivity) _ 1-1

Background to ODBC _____	1-3
Architecture _____	1-5
Driver Manager _____	1-6
Drivers _____	1-6
Data Sources _____	1-11
Conformance and Standards _____	1-13
ODBC Review Questions _____	1-15
ODBC Review Answers _____	1-16
Bibliography _____	1-18

Module 2. Introduction to SQL (Structured Query Language) __ 2-1

Structured Query Language _____	2-3
Tabular Structure of Data _____	2-3
Indexes _____	2-5
Combining Information from Different Tables _____	2-6
Additional Notes _____	2-8
Manipulating Data using SQL _____	2-9
SELECT statement basics _____	2-9
SELECT Statement - Advanced _____	2-11
INSERT Statement _____	2-18
UPDATE Statement _____	2-19
DELETE Statement _____	2-20
Summary _____	2-21
SQL Review Questions _____	2-22
SQL Review Answers _____	2-23
Further Exercises _____	2-24
Suggested Answers to Exercises _____	2-26
Additional Notes _____	2-28
The Easysoft Catalog _____	2-28
Supported SQL _____	2-28
Reserved Words _____	2-29
Bibliography _____	2-31
Easysoft Query for Windows _____	2-32
Start and Connect _____	2-32
The Example Database _____	2-35
Exercises _____	2-37
Suggested Answers _____	2-39

Module 3. Introduction to RMS Files	3-1
File Organisation Overview	3-3
File Attributes	3-5
Sequential Files	3-6
Relative Files	3-7
Indexed Files	3-9
Keys	3-9
Indexes	3-12
Record Access Mode	3-15
Record Format	3-17
File Protection	3-20
UIC-Based Protection	3-20
ACL-Based Protection	3-22
Summary	3-23
RMS Review Questions	3-24
CURRENCY_RATE.DAT	3-25
RMS Review Answers	3-26
Exercise Model Answers	3-28
Bibliography	3-30
Module 4. The Easysoft System (RMS)	4-1
Easysoft Architecture	4-3
Overall Process of using Easysoft	4-4
Data Sources and Catalogs	4-5
Defining Tables	4-6
Easysoft Catalog Structure	4-9
View the Catalog Tables	4-10
Accessing Server Data	4-15
Summary	4-16
Review Questions	4-17
Review Answers	4-18
Module 5. Microsoft ODBC Administrator	5-1
Purpose	5-2
Administrator v3.0	5-3
Summary	5-9
Supplement: troubleshooting	5-10

Module 6. Using ODBC Applications	6-1
Microsoft Excel	6-2
Download Entire Table	6-2
Add Criteria	6-5
View SQL	6-7
Return Data to Excel	6-8
Microsoft Access	6-10
Download Entire Table	6-10
Add Criteria	6-12
View SQL	6-14
View Indexes	6-14
Lotus 1-2-3	6-15
Configure Lotus 1-2-3 to work with Easysoft ODBC	6-15
Download Entire Table	6-16
Add Criteria	6-18
View SQL	6-19
Microsoft Word Mail Merge	6-20
Impromptu	6-24
Preparation: Create an Impromptu Catalog	6-24
Download Entire Table	6-27
Add Criteria	6-28
View SQL	6-29
Crystal Reports	6-30
Download Entire Table	6-30
Add Criteria	6-33
View SQL	6-34
Module 7. RMS Administration	7-1
Introduction	7-2
Determine File and Record Structure	7-4
Method 1 - Directory Command	7-4
Method 2 - Use FDL	7-5
FDL for CURRENCY_RATE.DAT	7-6
Determining Fields in a Record	7-7
Determine the Keys	7-8
Creating and Maintaining Files	7-9
Define a New FDL File	7-9
Tutorial: Create a New RMS Data File	7-18
Add a Further Key to an RMS Indexed File	7-19
Summary	7-20
Review Questions	7-21
Exercises	7-22
Review Answers	7-24
Answers to Exercises	7-25
Supplement: FDL File Structure	7-26
Description of KEY Secondary Attributes	7-27
Supplement: File Protection	7-30

Module 8. Easysoft Administration on the PC	8-1
Determine Field Lengths and Offsets	8-2
Stages of Defining a Server File	8-3
Starting the Administrator	8-5
Subsequent Administrator Logon	8-6
File Definition Tutorial	8-7
Step 1. Download Easysoft Catalog	8-8
Step 2. File Definitions	8-9
Step 3. Field Definitions	8-10
Step 4. Database Definitions	8-13
Step 5. Table Definitions	8-14
Step 6. Column Definitions	8-16
Step 7. User Definitions	8-17
Step 8. Create Database Privileges	8-18
Step 9. Upload Easysoft Catalog	8-19
Populate Server File	8-21
Importing and Exporting Definitions	8-23
Exporting Definitions	8-23
Importing Definitions	8-24
Summary	8-27
Review Questions	8-28
Review Answers	8-30
Supplement: Easysoft PC Administrator Installation	8-31
Download the Software	8-31
Administrator Installation Steps	8-31

Module 9. Easysoft Administration on the Server	9-1
Dealing with Catalogs	9-2
Create a Catalog	9-2
Changing Catalog Passwords	9-3
Exporting a Catalog	9-4
Importing a Catalog	9-5
OpenVMS Passwords and Easysoft	9-6
Easysql	9-7
Easysoft COBOL Converter	9-8
Summary	9-9
Review Questions	9-10
Conversion Exercise	9-11
Exercise Steps	9-12
Review Answers	9-15

Module 10. RMS Training Data	10-1
Easysoft Travel Company	10-2
Description of RMS Data Files	10-3
SQL Tables	10-5
RMS Training Data	10-6
FDLs for RMS Files	10-12
CSV Description	10-21
Module 11. RMS Exercises	11-1
General Information	11-2
Tips and Reminders	11-2
Task 1 : Produce a list of all Suppliers	11-4
Task 2 : Produce a List of Supplier Names	11-5
Task 3 : Produce a directory of all Suppliers	11-6
Task 4 : Produce a directory of Virgin Holidays Ltd (S0012)	11-7
Task 5 : Produce a list of addresses of the Suppliers in London	11-9
Task 6 : List all Spanish Destinations	11-10
Task 7 : List CUSTOMERS with the name Smith or Jones	11-11
Task 8 : List the Brochures and the Suppliers they belong to	11-13
Task 9 : List all brochures for Thomson Holidays Ltd	11-15
Task 10 : List existing currencies and currency rates	11-17
Task 11 : Find the currency used in Cancun	11-21
Task 12 : How many customers are there?	11-23
Task 13 : Update the CUSTOMER file	11-24
Task 14 : Insert a New Supplier Record	11-25
Task 15 : Delete a Supplier record	11-26
Task 16 : Find the Holiday company we have spent the most money with	11-27
Task 17 : Find the most popular holiday destination	11-30
Task 18 : Prepare a report to show the Total Profits	11-32
Task 19 : Prepare a letter to all Clients who spent over £5000.00 in May	11-34
Task 20 : Produce a Purchase Invoice form	11-36
Task 21 : Insert data into an RMS file from local Access Tables	11-42
Module 12. Easysoft Excel Macro for RMS	12-1
Overview of Macro	12-2
Macro Installation	12-3
Initialisation	12-5
Download Data	12-7
Complex Criteria	12-11
Pivot Table	12-13
Upload Data	12-16
Report	12-18
Setup New Report	12-18
Run Report	12-19
Advanced Techniques	12-19
Troubleshooting	12-22

Appendix A. Global Glossary	3
Appendix B. Data Types	10
SQL Data Types	10
Local Data Types	11
Appendix C. Import Export CSV Format	15
Appendix D. Easysoft ODBC PC Installation	16
Starting the Driver Installation	16
Appendix E. Easysoft COBOL Converter	20
Installation	20
Using the Converter	21
Basic Conversion	22
Output FILLER Fields	24
Output Group Items (sub-structured fields)	24
Convert Raw Record Descriptions	25
Notes / Information	26
COBOL Data Types	27
Appendix F. Easysoft PowerHouse PDL Converter	30
Using the Converter	30
Basic Conversion	32
Output FILLER Fields	33
Output Sub-structured Fields	34
Stripping Characters	35
Define Conditions for use by the Pre-processor	35
Notes / Information	36
Appendix G. Troubleshooting	37
Easysoft ODBC Logging	38
EASYSOFT.INI	39
Microsoft Trace Options	41
Version 3.0 Trace Options	41
SQL.LOG Described	42
How to Work Out What the Server is Doing	43
Find the Process Causing the Problem	43
View Log File	45
Stop the Process	45
Appendix H. Using Easysoft Support Services	47
Support Check List	48
The Easysoft FTP Site	49
Sending Data to Easysoft	50
Appendix I. RMS in Context	51
Overview of OpenVMS	53
Digital Command Language	56

1. Introduction to ODBC (Open Database Connectivity)

This module outlines the Open Database Connectivity (ODBC) functionality and architecture. These course notes address the background problem that ODBC deals with.

In this module you will learn

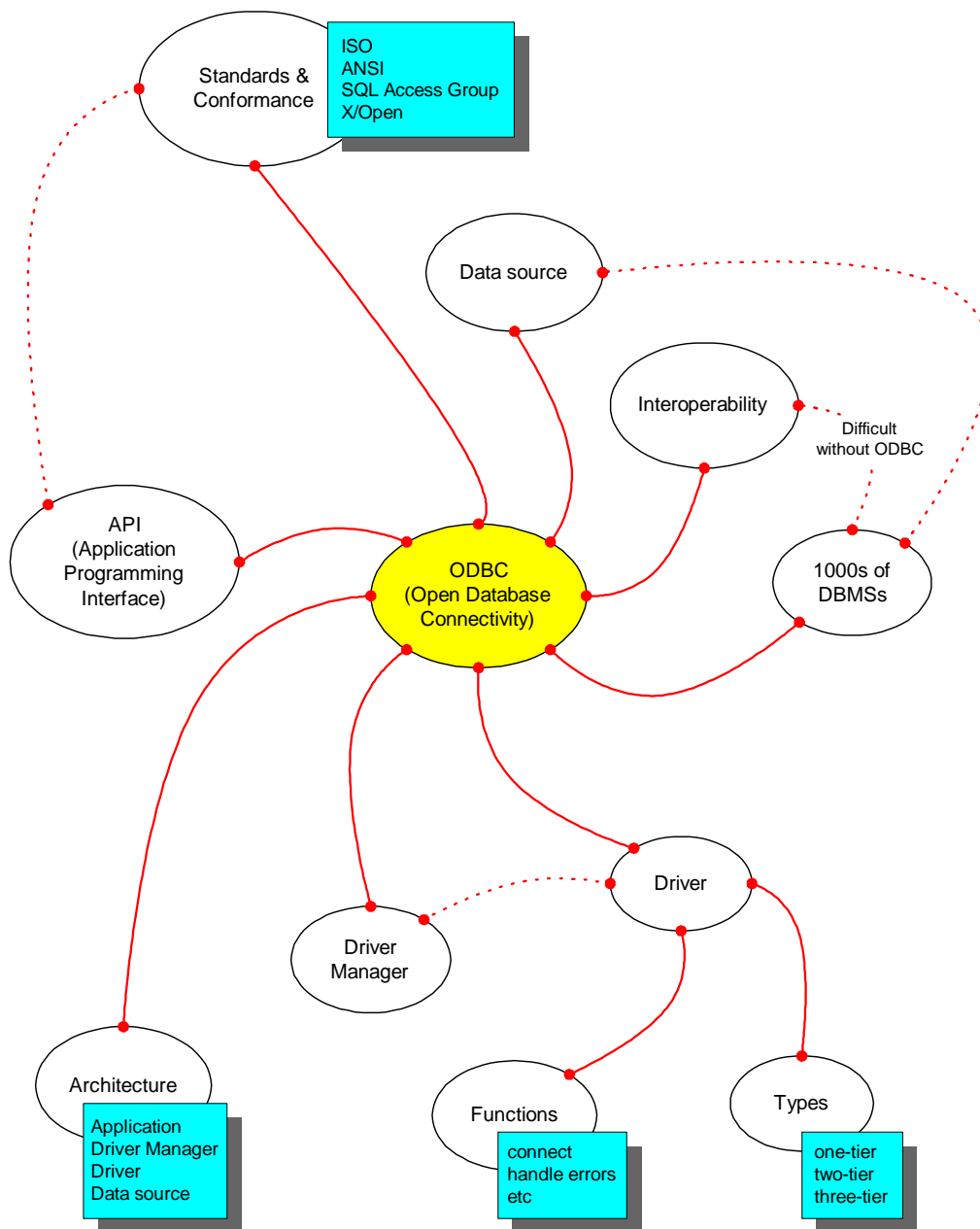
- ✦ why we need ODBC and what the alternatives are
- ✦ the architecture of ODBC
- ✦ the terminology used with ODBC

At the end of the module there are some review questions.

Contents

Background to ODBC _____	1-3
Architecture _____	1-5
Driver Manager _____	1-6
Drivers _____	1-6
Data Sources _____	1-11
Conformance and Standards _____	1-13
ODBC Review Questions _____	1-15
ODBC Review Answers _____	1-16
Bibliography _____	1-18

ODBC Mind Map



Background to ODBC

This section presents a brief background to ODBC, and then we discuss the overall architecture of ODBC, the Driver Manager, drivers and conformance.

Position

Hundreds of different software manufacturers produce *Database Management Systems* (DBMS) for storing, accessing and manipulating data. For various reasons, users may wish to access data that is stored using any number of different DBMSs via various PC applications.

Problem

Typically, each different DBMS uses storage and access methods for the *database* (i.e. collection of data files) that are incompatible with any other DBMS, thus complicating data access via PC applications.

Alternatives

1. Each PC application contains different versions of software to enable it to communicate with all the DBMSs on the market.

For example, consider Excel and three different Database Management Systems, such as Oracle, Ingres and Sybase.

It would take three different versions of Excel - for Oracle, Ingres and Sybase - and a vast amount of work from developers of these applications. Plus, users would have to buy and maintain three different versions of Excel.

2. Define a standard that uses common terms to describe the data and operations on that data.

For each different DBMS, the software vendor provides software which converts the common terms into the particular operations of the DBMS. Conversely, any outgoing data is presented in the defined format.

Any computer program, providing it conforms to the standard, can send a request to the DBMS, and the data will be returned in a standard format.

Solution

The latter option is the ODBC solution to the problem. ODBC was developed by Microsoft to give a single *API* (Application Programming Interface) which can access a variety of data sources. This means developers can write applications which are not targeted to any specific DBMS. End users connect applications to their databases by using add-in modules called *drivers*, which are available from database vendors and driver developers.

The aim of ODBC is to allow interoperability between different DBMSs and applications which run under the Microsoft Windows operating system. It works by supporting heterogeneous data access - in other words, applications access different data sources by using drivers which access the data. Applications can submit any SQL statement which is supported by a given driver.

SQL is a computer language that is used with a particular type of database called a relational database, and in general, when reference is made to databases, we mean relational databases. It is possible to use ODBC, in conjunction with additional software, to access non-relational data. This is exactly what we do at Easysoft.

GLOSSARY

API (Application Programming Interface) A set of standards that allows an application to connect to a DBMS.

Database A collection of data files.

Data Source (in ODBC terms) A combination of the data, the DBMS used to access that data, the operating system and the network.

DBMS (Database Management System) The software that manages access to a database.

Driver The software that implements ODBC functions.

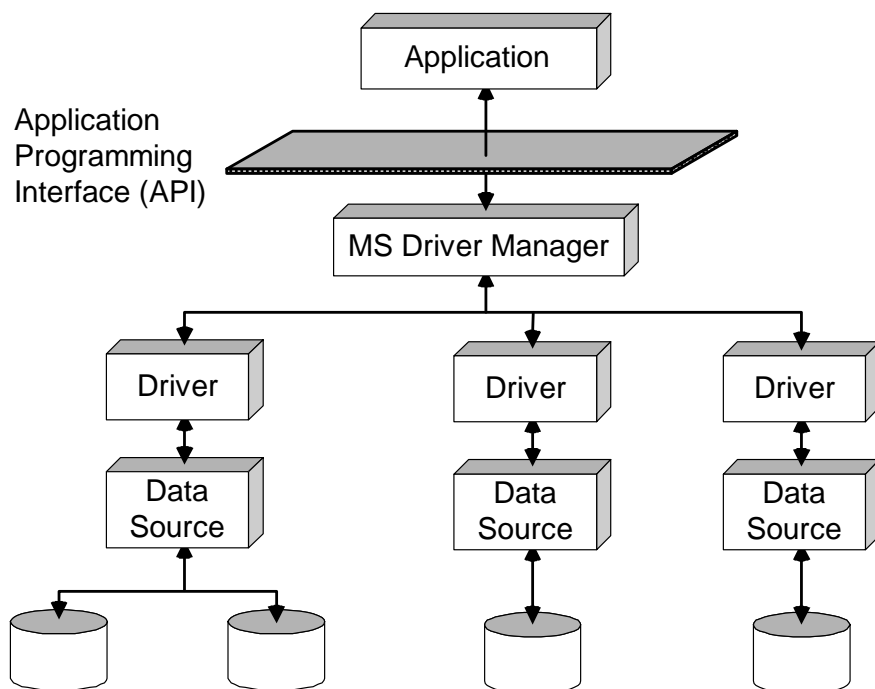
SQL (Structured Query Language) An official standard computer language for interacting with (relational) database systems.

Architecture

The four main components of the ODBC architecture are the *Application*, the *Driver Manager*, the *Driver* and the *Data source*. Their relationship is shown below, and their functions are briefly described. The following section then explain the components in more detail.

Note: in later modules you will use the Microsoft ODBC Data Sources Administrator. For all practical purposes, this is equivalent to the Driver Manager.

ODBC Architecture



The ODBC Interface, which essentially defines the API and SQL syntax, is not part of the architecture, and is described in the section entitled “Conformance” on page 1-13.

An application, such as Microsoft Access or Lotus 1-2-3, calls ODBC functions (i.e. requests the performance of ODBC operations) which are sent to the driver via the Driver Manager. As far as the application is concerned the driver and Driver Manager appear to be a single functional unit. A single application may wish to access data from a number of different sources and ODBC is designed to allow this.

The Driver Manager (provided by Microsoft) loads drivers for an application when an application calls certain SQL functions and, if requested, traces calls and keeps a log file of these.

The driver processes the function calls sent by the application, submits SQL calls to the data source and returns results to the application. It may change the syntax of the request to conform to the syntax used by the DBMS in the data source. Another function of the driver is to format errors into standard error codes and return these codes to the application.

In the context of ODBC a data source is more than just a set of data files. A data source consists of a set of data and its associated environment, which includes the operating system, Database Management System and networks (if any). Typically, the term data source is used in a loose fashion to describe any software component that is not one of the other three described above. An application may access more than one data source at any given time if it so requires.

Driver Manager

The Driver Manager sits between applications and drivers in the ODBC architecture and it manages the interactions between an application and a driver. Both multiple applications and multiple drivers can be managed simultaneously by the Driver Manager.

An application calls an ODBC function, and the Driver Manager sends that call to the appropriate driver. The first time that an application uses ODBC to connect to data, the required driver is determined by the Driver Manager and is then loaded into memory. Thereafter, the Driver Manager takes each incoming function call from the application and calls a function of the same name in the driver. (The only exception to this is when the function call is one that the Driver Manager is expected to process, such as when an application asks for the name of a driver). Finally, when the application calls the ODBC function which requests disconnection, the Driver Manager unloads the driver from memory (unless another application is also using that driver).

Additionally, elementary error checking is performed by the Driver Manager to ensure that functions are called in the correct order and that arguments contain valid values.

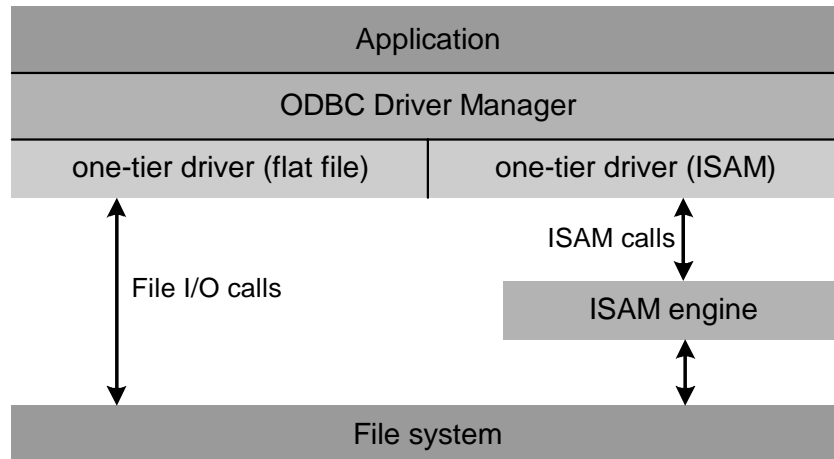
Drivers

First, we look at the architecture of drivers, and then the basic functions of a driver are presented.

Types of Driver

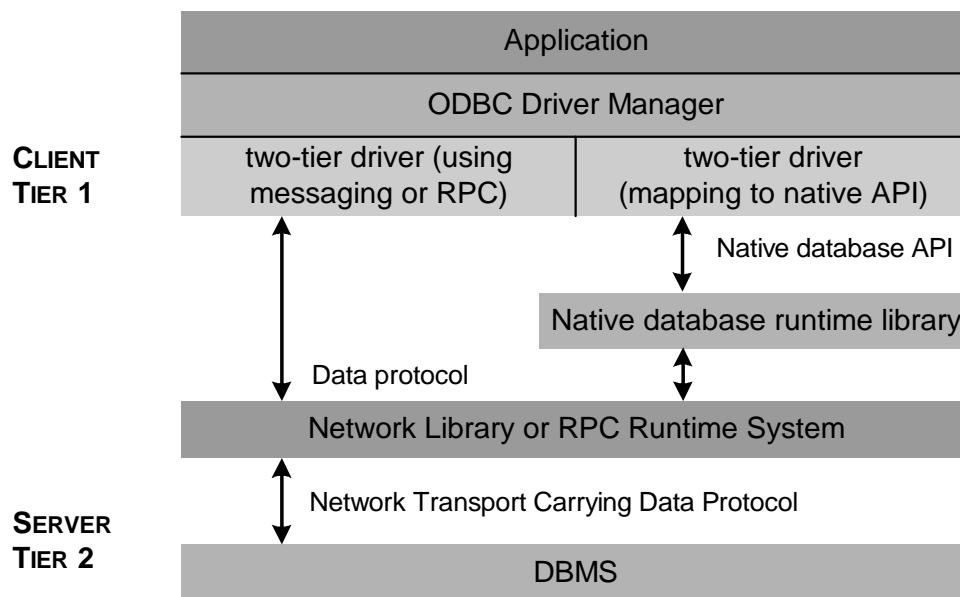
Two basic types of driver are defined by ODBC. A *single-tier driver* processes data directly (i.e. it processes both ODBC calls and resultant SQL statements) whereas a *multi-tier driver* sends SQL statements to the data source. Multi-tier drivers can be further described as two-tier drivers and three-tier drivers. This is shown in the figures below.

One-tier driver architecture



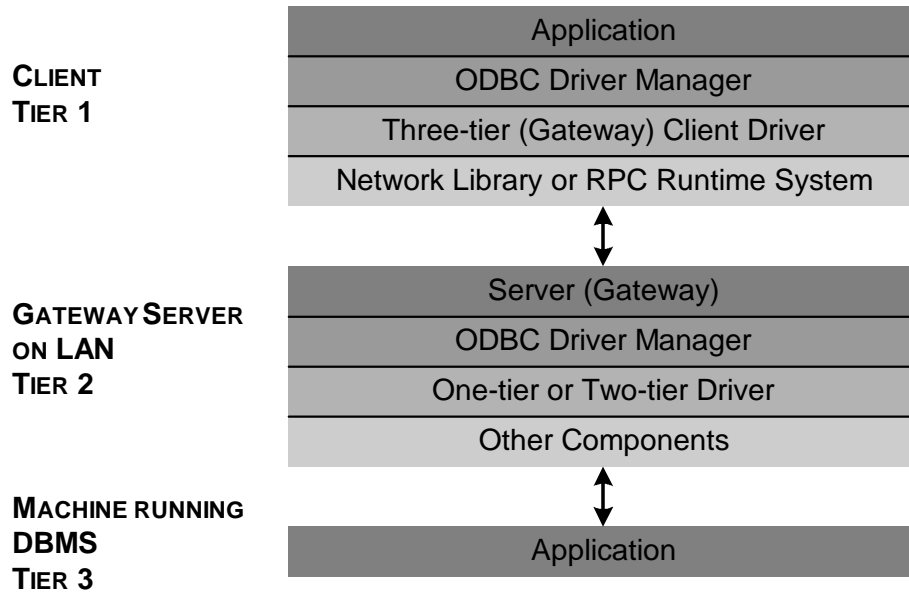
The term one-tier in ODBC refers to a driver that accesses a desktop database file or a flat file. The main distinguishing feature of one-tier drivers is that the driver itself does all of the SQL processing.

Two-tier driver architecture



Two-tier systems are the classic client/server systems. The client, i.e. the driver, sends and receives the Database Management System's data protocol or maps to the native database API. It doesn't access the data directly. The server, i.e. the DBMS, receives SQL requests from the client, executes them and then sends the results back to the client.

Three-tier driver architecture



From the perspective of a client, there is little difference between two-tier and three-tier drivers. The main difference is that in a three-tier system, the client connects to a server which acts as a gateway to the target Database Management System.

To avoid confusion it is important to note the distinction between driver architecture and hardware architecture. A single-tier driver can run on a physical client-server hardware architecture (i.e. the driver accesses data that is stored on a server, but the driver itself still sends SQL from the client to the server). Conversely, a multi-tier driver may reside on a single machine - the driver sends SQL requests to the data access software which processes these requests.

Typically, however, in a multi-tier configuration, the application, driver and Driver Manager reside on one machine (the client), and the database and data access software reside on a different system (the server). This has the advantage over a single-tier driver on client-server hardware that only the required data is sent over the network.

Driver Functionality

Once a driver has been loaded by the Driver Manager, there are a number of tasks it performs:

- Connect to machine containing data
- Deal with errors
- Transform SQL
- Catalog functions
- Information functions
- Data type conversions

Connect to Machine Containing Data

Once the Driver Manager has loaded a driver, the driver first connects to the machine on which the data resides. For one-tier drivers there is no network communication to deal with because the data is local. Two-tier and three-tier drivers must also deal with network communication.

Consider a two-tier driver. It has to establish a network connection to the DBMS. Typically, the driver is responsible for displaying a dialog box in which users type their names and passwords for logging into the DBMS. Then, after this security check, the driver loads the correct network library (unless it is already linked into the driver). The network library functions then connect to the server using the server name or network address information that was provided when the data source was set up.

Deal with Errors



To give applications a standard way of dealing with error conditions, ODBC requires drivers to provide not only DBMS-specific error codes, but also standard error codes. Although the overall types of errors that are returned by DBMSs are similar, the internals, such as error numbers, message content and the amount of information displayed will all be different. To overcome these differences ODBC provides:

- a return code mechanism which reports success or failure for each function
- a standard error function that applications can call every time an ODBC function fails or presents a warning
- standard error codes for over 85 error conditions
- all the error information that is supplied from the DBMS; native error codes and messages are returned to the application
- error information from the driver itself
- a tagging scheme that identifies the component that reported the error. This example shows the Easysoft driver returning a server login error message:



- the ability to return multiple errors from a single ODBC function call

From the perspective of an end user who is using an interactive application, the main advantage to the standard error handling is the common style. However, if an application itself makes use of the errors returned from a DBMS in order to perform some further action, then there is an important advantage to having a standard code. Instead of needing programming code to deal with all the possible DBMSs that might be used by the application, there only needs to be one check for the standard error that is returned.

Transform SQL

DBMSs do not always use standard SQL as defined by ISO/ANSI, and therefore, drivers need the capability of transforming one dialect of SQL into another. Some DBMSs do not provide all the functionality defined in the standard, and drivers need not need to add functionality. However, if a DBMS provides the same functionality as the standard, but if it does it differently from the standard, then the driver must translate the standard into a form that is understood by the DBMS.

ODBC provides extensions to standard SQL which provide essential interoperability functionality. For example, many DBMSs support the SQL data types DATE, TIME and TIMESTAMP. However, the literal formats used with SQL statements can vary between DBMSs. The ISO SQL standard representation for DATE data is *yyyy-mm-dd*, where *yyyy* represents four digits corresponding to the year, *mm* represents two digits corresponding to the month and *dd* represents two digits corresponding to the day. This is what ODBC uses; if a driver supports the DATE data type, then it must transform this into the syntax of the DBMS that it is accessing.

Catalog Functions

A major task of a driver is to provide information about the tables, columns and other objects in a DBMS in a uniform way so that users can interactively select the items of interest. ODBC provides a set of catalog functions which provides applications with the basic information that they need from the catalog in the DBMS. Each driver generates an SQL query that returns the required information to the application. The catalog functions defined in ODBC include the ability to return information about:

- tables that a user can access (*SQLTables* function)
- the columns in a table (*SQLColumns* function)
- information about a table, such as indexes, if there are any, that are defined on the table (*SQLStatistics* function)

Information Functions

A driver must be able to provide information about its capabilities and the capabilities of the DBMS it interacts with. A few important functions are shown here:

- *SQLGetInfo* informs the application of the conformance level, version and functionality capabilities of the driver

- *SQLGetTypeInfo* returns information about the data types a DBMS uses.
- *SQLDataSources*, *SQLDrivers* and *SQLGetFunctions* inform the application of all the available data sources, all the drivers and all the functions that a driver supports.

Data Type Conversions

One of the goals of ODBC is to provide data in a form that is most suitable to any given application.

Any data type conversion that makes sense is allowed by ODBC. The application specifies the conversion, and the driver must do this conversion.

Data Sources

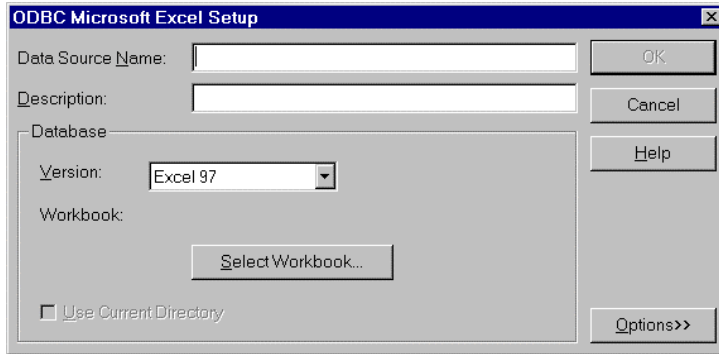
Ideally, users should not be aware of underlying communications software, the drivers, the DBMSs, server addresses etc. To hide these things from end users, ODBC uses a *data source name* (DSN). All the underlying software components of any given data source are mapped to one data source name. Typically, the data source name is chosen by the end user or a system administrator, and ideally, it should be a name that self-evidently describes the data that it represents (for example, "SALES DATA").

Within ODBC the term *data source* is used for two purposes:

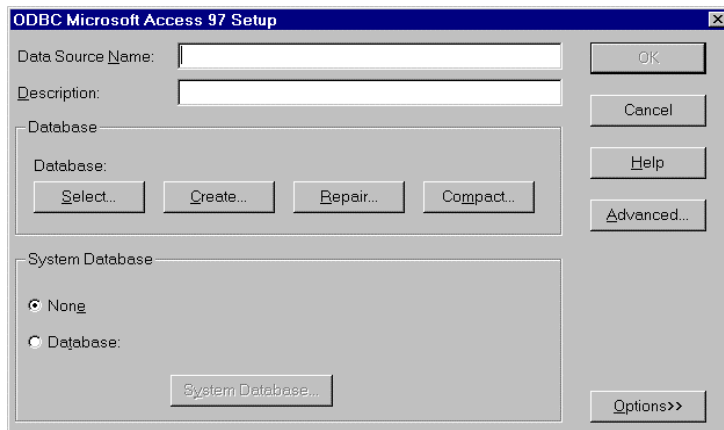
1. It defines the data that an end user wants to access. The data source includes the DBMS software that runs on a particular machine, the operating system, and networking software.
2. It is used to refer to the data source name (so instead of saying, for example, "connect to the data source which is defined by the 'SALES' data source name", we say "connect to the 'SALES' data source").

Every driver has its own particular set up, and a few examples are shown here.

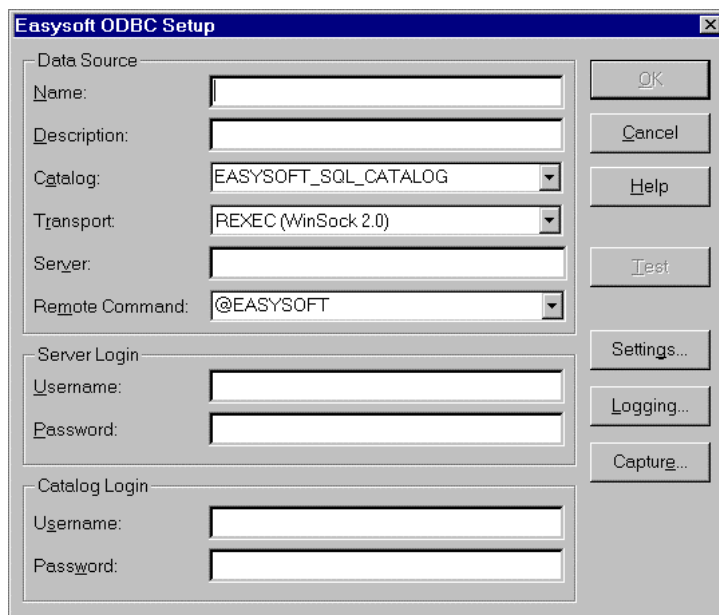
ODBC Microsoft Excel Setup



ODBC Microsoft Access 97 Setup



Easysoft ODBC Setup



Conformance and Standards

Successful operation of ODBC depends upon defining a set of standards common to all participating elements of a system.

ODBC is used to facilitate data communication between different applications. It allows the use of plug-in modules - in other words, drivers from many different vendors can be used. However, all DBMSs and applications provide different functionality and so for systems to communicate, it is necessary to define standards for functionality. ODBC defines conformance levels in two areas of functionality, the ODBC API and SQL grammar (including data types), and for both of these there are three levels of conformance, each one being more comprehensive than the level below.

The conformance levels for the API are named: core API, level 1 API, level 2 API.

The conformance levels for SQL are named: minimum SQL grammar, core SQL grammar, extended SQL grammar.

Selecting an appropriate level of conformance depends upon the needs of the application. Microsoft suggest that driver developers implement all level 1 ODBC API functions since many ODBC applications require this. Conforming to a given level does not mean that additional functionality cannot be provided, but if a driver is claimed to conform to a given level, then all the functionality of that level should be provided.

Conformance levels can be determined in a number of ways:

- from driver documentation
- call the *SQLGetFunctions* function from the Driver Manager
- if a driver supports the *SQLGetInfo* and *SQLGetTypeInfo* functions you can use these to return information on conformance. Although these are level 1 functions many core level drivers support them

Standards Groups

Various standards groups have influenced the development of ODBC. Notable ones are

ANSI (American National Standards Institute) This is a U.S. government agency which sets standards in commerce and industry. It works in collaboration with the ISO (below).

ISO (International Standards Organisation) This is a federation of national standards organisations.

SQL Access Group. This was formed in 1989 with the aim of accelerating the acceptance of formal standards which would be use to increase portability and interoperability for database applications. The SQL Access Group merged with X/Open (below) in 1994.

X/Open. This organisation deals with portability issues in a practical way. Wherever possible, X/Open follows ISO standards, but if such standards do not exist, then this group will define additions to existing standards in order to help developers create portable applications.

Summary

In this module you have learnt

- ✦ ODBC allows different applications to communicate by means of a common interface
- ✦ Applications use plug-in modules called drivers to communicate
- ✦ When an application wants data, it connects to a data source using a driver
- ✦ The Microsoft Driver Manager loads drivers for a calling application
- ✦ A data source consists of a set of data and its associated environment, which includes the operating system, Database Management System and networks (if any)
- ✦ Single-tier drivers process both ODBC calls and SQL statements (SQL is a language for relational databases)
- ✦ Two-tier and three-tier drivers send SQL to the data source
- ✦ Three conformance levels are defined for both ODBC and SQL

ODBC Review Questions

1. What does the acronym ODBC stand for?

2. What is ODBC used for?

3. The four main components of ODBC architecture are:
 - a.
 - b.
 - c.
 - d.

4. What is a database?

5. Name at least two functions of the Microsoft Driver Manager

6. What is the function of a driver?

7. Name the three basic types of driver architecture.

8. What's the difference between the three driver architectures?

9. What is Structured Query Language (SQL)?

10. What is an API (Application Programming Interface)?

11. What is the function of conformance levels?

ODBC Review Answers

1. What does the acronym ODBC stand for?

Open Database Connectivity

2. What is ODBC used for?

To connect different (and hence incompatible) databases

3. The four components of ODBC architecture are:

a. Application b. Driver Manager c. Driver d. data source

4. What is a database?

A collection of data files

5. Name at least two functions of the Microsoft Driver Manager.

a. loads drivers
b. traces and keeps a log of SQL function calls
c. list the driver that is used by each data source (information is taken from the ODBC.INI file)

6. What is the function of a driver?

a. Processes ODBC function calls which are sent by an application
b. Submits SQL calls to a data source
c. Returns results to an application
d. Format errors into standard forms, and return these errors to an application

7. Name the three basic types of driver architecture.

a. Single-tier
b. Two-tier
c. Three-tier

8. What's the difference between the three driver architectures?

A one-tier driver processes data directly (i.e. it processes both ODBC calls and the resultant SQL statements).

Two-tier and three-tier drivers send SQL statements to the data source for processing. The main difference between two-tier and three-tier drivers is that with three-tier drivers the client connects to a server which acts as a gateway to the DBMS.

9. What is Structured Query Language (SQL)?

An official standard language for interacting with relational database systems.

10. What is an API (Application Programming Interface)?

An API is a set of standards that allows an application to connect to a DBMS.

11. What is the function of conformance levels?

Since all DBMSs and applications provide different functionality, in order for systems to communicate, it is necessary to define standards for functionality.

Bibliography



This bibliography lists a few sources of information relevant to information in this module. Entries are listed alphabetically by title, except that initial articles (i.e. 'A' and 'The') are ignored for the purposes of ordering.

Inside ODBC, K. Geiger. Published by: Microsoft Press, 1995. ISBN 1-55615-815-7.

Microsoft ODBC 2.0 Programmer's Reference and SDK Guide, Microsoft. Published by Microsoft Press, 1994. ISBN 1-55615-658-8.

Microsoft TechNet CD. This CD contains technical information relevant to the Microsoft Corporation. The CD is updated frequently.

Teach yourself ODBC Programming in 21 days. B. Whiting, B. Morgan, J. Perkins. Sams Publishing, 1996. ISBN 0-672-30609-3.

The ODBC Solution - Open Database Connectivity in Distributed Environments, R. Signore, J. Creamer, M. O. Stegman. Published by McGraw-Hill, 1995. ISBN 0-07-911880-1.

Using ODBC 2, R. Gryphon, L. Charpentier, J. Oelschlaeger, A. Shoemaker, J. Cross, A. W. Lilley. QUE Corporation, 1995. ISBN 0-7897-0015-8.

2. Introduction to SQL

(Structured Query Language)

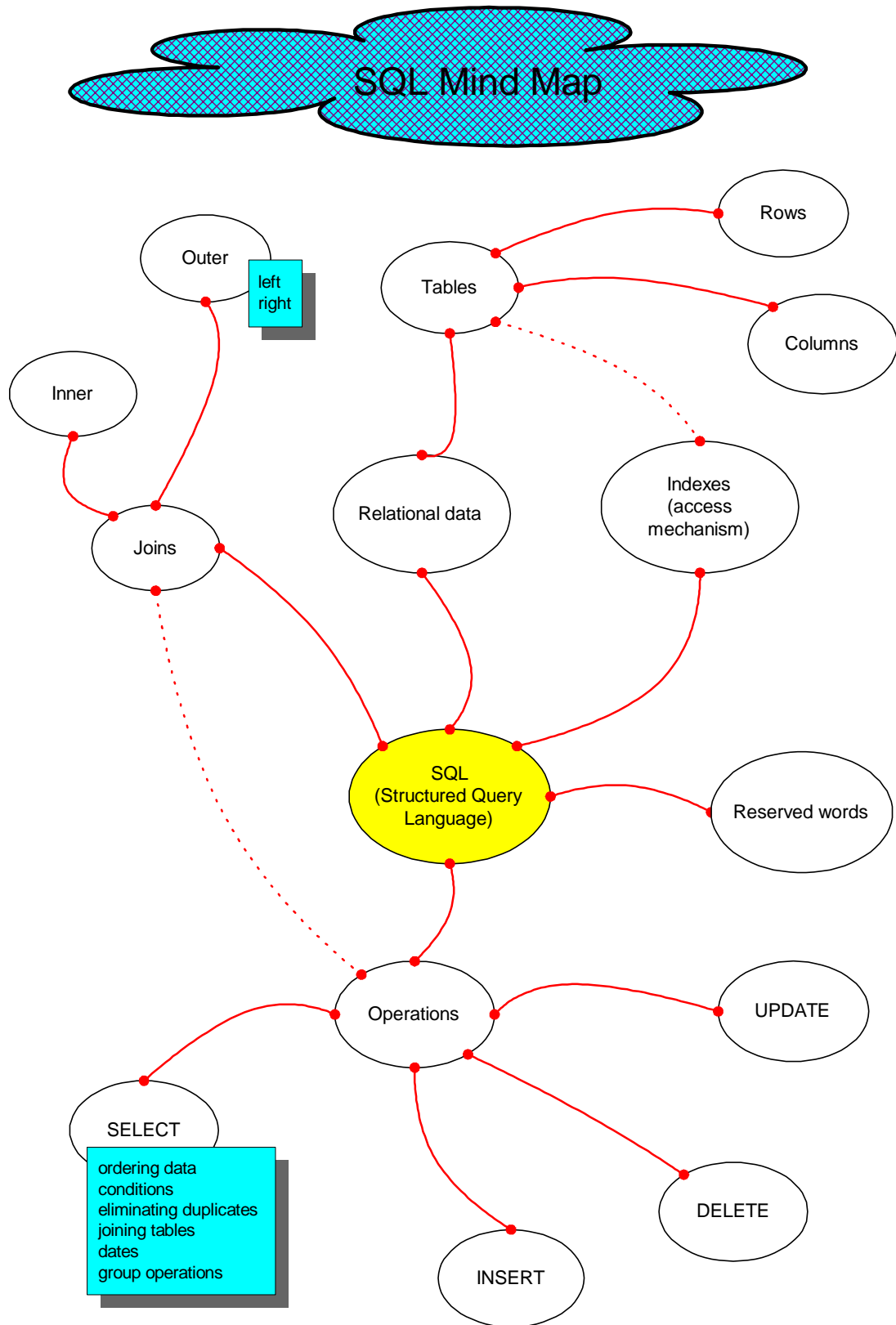
In this module you will learn

- ✧ what SQL is and the basic properties of SQL
- ✧ the tabular structure of data
- ✧ how to retrieve, enter, change and delete data using SQL
- ✧ how to set conditions to specify what data is returned in a query
- ✧ how to obtain data from more than one table
- ✧ how to perform aggregate functions (e.g. count the records, sum of values) on the data that is returned

At the end of the module there are some review questions.

Contents

Structured Query Language	2-3
Tabular Structure of Data	2-3
Indexes	2-6
Combining Information from Different Tables	2-7
Additional Notes	2-9
Manipulating Data using SQL	2-10
SELECT statement basics	2-10
SELECT Statement - Advanced	2-12
INSERT Statement	2-19
UPDATE Statement	2-20
DELETE Statement	2-21
Summary	2-22
SQL Review Questions	2-23
SQL Review Answers	2-24
Further Exercises	2-26
Suggested Answers to Exercises	2-28
Additional Notes	2-30
The Easysoft Catalog	2-30
Supported SQL	2-30
Reserved Words	2-31
Bibliography	2-33
Easysoft Query for Windows	2-34
Start and Connect	2-34
The Example Database	2-37
Exercises	2-39
Suggested Answers	2-41



Structured Query Language

SQL stands for *Structured Query Language*. It is a computing language for interacting with *relational* database systems. In simple terms, a relational database is one in which all the data is perceived as being contained in *tables*, and a table itself is composed of rows and columns of data.

SQL is used with ODBC because ODBC is designed for relational databases. The idea behind ODBC is that any relational database can communicate with any other relational database or application. The differences in physical implementation between different database systems are overcome because ODBC defines standards for communicating - any application that supports ODBC (i.e. an ODBC-compliant application) can send and receive data from any other application that conforms to ODBC standards. There are different levels of conformance to various ODBC standards and there are official standards for different versions of SQL. In addition, there are ODBC extensions to SQL. For the discussion here, these different standards are not important, but we do sometimes use the ODBC extensions (these are clearly indicated when they are used).

SQL began to be developed in the early 1970s by IBM and has grown considerably since then. By the late 1980s, both the American National Standards Institute (ANSI) and the International Standards Organisation (ISO) had produced definitions for SQL. In the business and industrial world there are many hundreds, probably thousands, of applications and systems which use SQL.

Tabular Structure of Data

All data is perceived to be stored in *tables*. These consist of *columns* and *rows*. A column is the vertical dimension of the table and a row is the horizontal dimension of the table (sometimes columns are also called fields and rows are called records). A *database* is a collection of tables.

Imagine that there is a database that is used to store information about both the products that are made by a company and the orders that customers send to the company. For now, consider only the products that are made. Say that we need information on only the product number, name and price. This information is shown in the PRODUCTS table below.

PRODUCTS		
P_NUMBER	P_NAME	P_PRICE
P1	widget	13.50
P2	fergulator	19.60
P3	doofer	15.45
P4	dongle	8.05

In this table there are four rows corresponding to the four products made by the company. For each of these products, three items of information relating to that product are stored, namely the product number, the product name and the price.

Say that we want to keep information about orders despatched. For this, we will use two tables. First, here is the ORDERS table, which contains information about each order.

ORDERS		
O_NUMBER	CUSTOMER	O_DATE
O1	Smith	29-01-1997
O2	Jones	29-01-1997
O3	Black	03-02-1997
O4	Smith	04-02-1997

The format of date information in a DBMS (Database Management System) is application dependent.

For exemplary purposes we don't consider any further the relationships between other tables that would be required in a real-world case. Let us say that the CUSTOMER column sufficient to identify all the customers known to the database.

For each order, we need to know the product(s) requested, and the number required. There is another table which is used to keep track of the details of each order that are despatched by the company. This is called OLINES (for order lines).

OLINES			
O_NUMBER	P_NUMBER	QUANTITY	TOTAL_PRICE
O1	P1	1	13.50
O1	P4	2	16.10
O3	P4	1	8.05
O4	P2	1	19.60
O4	P4	1	8.05

To relate information in different tables, the tables must be "joined" (see "Combining Information from Different Tables", page 2-7 of this module).

GLOSSARY

Column The vertical dimension of a table.

Database A collection of tables.

Row The horizontal dimension of a table.

Table A table contains data which is arranged in rows and columns.

Indexes

To speed up the rate of data retrieval, indexes can be used. An index can be unique or non-unique. Indexes provide ordered access to rows of a table, and they are based on the values of one or more columns of a table. An additional feature is that the ordering may be ascending or descending.

Although indexes are useful when retrieving data, they slow down the input of data (because in addition to updating the table, the index must be maintained). It is not possible to state definitively when indexes should and should not be used, because this depends upon how the data is used. Typically, if the data does not change frequently, it will be more beneficial to use an index than if the data changes frequently.

As example of a unique index, consider the ORDERS table. This could have a unique index defined on the O_NUMBER column. If we are looking for a specific order, then this index would allow rapid retrieval of the data.

ORDERS		
O_NUMBER	CUSTOMER	O_DATE
O1	Smith	29-01-1997
O2	Jones	29-01-1997
O3	Black	03-02-1997
O4	Smith	04-02-1997

unique index: every data value in this column is different.

non-unique index: data values in this column can be the same.

A non-unique index could be defined on the CUSTOMER column, so that all the orders for a specific customer could be found rapidly.

Combining Information from Different Tables

Say we want to gather information for the order line details of a prototypical invoice. If we don't need to show part names, then we could just obtain data from the OLINES table. If we wanted to include part names, then we would have to combine information from both the OLINES and PRODUCTS tables, because product names are stored in the PRODUCTS table. This operation is called a join. The tables are joined on the P_NUMBER field, which is common to both tables.

Types of Join

There are essentially two basic types of join called *inner join* and *outer join*. The inner join selects records which match in both tables. The outer join operation selects all the records from one table, even if they do not have a corresponding match in the other table. Outer joins can be classified as left outer joins and right outer joins. A left outer join is one where all the records in the "left" table (i.e. the first table in the join condition) are shown. A right outer join is one where all the records in the "right" table (i.e. the second table in the join condition) are shown.

Inner and outer joins are exemplified using the OLINES and PRODUCTS tables. Say the join is OLINES joining PRODUCTS, and say that we just want to see just the P_NUMBER and P_NAME fields (purely so that this page is not cluttered with unnecessary text). For ease of reference, the relevant data is replicated here.

OLINES		PRODUCTS	
<u>O_NUMBER</u>	<u>P_NUMBER</u>	<u>P_NUMBER</u>	<u>P_NAME</u>
O1	P1	P1	widget
O1	P4	P2	fergulator
O3	P4	P3	doofer
O4	P2	P4	dongle
O4	P4		

On all the joins that will be shown, we will join P_NUMBER from the OLINES table with P_NUMBER in the PRODUCTS table. In reality, these P_NUMBERs are the same thing - the product number of a particular product. However, in our representation of reality, they are different, and we need some method for distinguishing them. The convention is to prefix the column name with the table name, so we have OLINES.P_NUMBER and PRODUCTS.P_NUMBER.

GLOSSARY

- Index** A data structure that behaves like an ordered list of pointers to the rows of a table.
- Inner Join** Returns records which match in the join column(s) in both tables.
- Join** Basically, a join is a query in which data is retrieved from more than one table.
- NULL** represents a case where a value is unknown.
- Outer Join** All records from one of the tables are returned, even if there are no matching values in the join column(s) of the other table.

Inner Join

The result of an inner join (just showing P_NUMBER and P_NAME, and with duplicates removed) is:

OLINES.P_NUMBER	PRODUCTS.P_NAME
P1	widget
P2	fergulator
P4	dongle

Of all the products available, only three products, namely P1, P2 and P4, appear in the OLINES table. Therefore these three products appear in the result of the inner join.

Left Outer Join

The 'left' table is OLINES, so all the product numbers which appear in this table will be matched with all the product numbers in the PRODUCTS table. The result is:

OLINES.P_NUMBER	PRODUCTS.P_NAME
P1	widget
P2	fergulator
P4	dongle

In our case, the result is the same as an inner join (because there cannot be any products in OLINES which do not occur in PRODUCTS).

Right Outer Join

The ‘right’ table is PRODUCTS, so all the product numbers which appear in this table will be matched with all the product numbers in the PRODUCTS table. The result is:

OLINES.P_NUMBER	PRODUCTS.P_NAME
P1	widget
P2	fergulator
NULL	doofer
P4	dongle

The product ‘doofer’ (P3) exists in the PRODUCTS table, so it appears (indirectly as P_NAME) in the join result. P3 does not exist in the O_LINES table, so it does not appear in the result. The term NULL in SQL refers to unknown or missing information.

Additional Notes

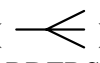
Typically, table names and the column names are typed in upper case. This is not mandatory, but is a convention - lowercase letters are allowed. Table and column names must start with an alphabetic letter and cannot contain spaces - the underscore character (_) is usually used as a separator.

SQL is not concerned with the actual storage of data.

There are many pictorial methods of describing the structure of a database. If just the table structure is being defined, then a useful method is to use a variant of a Bachman diagram; tables are shown as rectangles, and lines linking the rectangles show the relationships between tables. Our database would be shown thus:

Pictorial representation of database



The “crow’s foot” () indicates a one-to-many relationship, the single line at PRODUCTS and ORDERS representing the “one”, and the triple lines (at ORDERLINES) representing “many”. That is, one product can appear in many different order lines, but each order line can reference just a single product. Similarly, each order in ORDERS can have many (one or more) different lines, whereas any one order line belongs to just one order.

Manipulating Data using SQL

Although SQL is capable of far more than just manipulating data, as far as we are concerned, its purpose is to retrieve and store data. The four operations that are available for this are the SELECT, INSERT, UPDATE and DELETE statements. These do exactly what their names suggest, and we will look at each of them in turn, starting with the most complex (in terms of its richness of structure), namely SELECT. Each of the sections will start with a brief overview of the structure of the statement, followed by one or more examples from the example database described above. All the possible options for the statements are not covered - refer to one of the many books on SQL for a comprehensive reference. In the real world, SQL is often used to deal with millions of records; the principles can be explained best when there are only a few rows in each of the tables.

SELECT statement basics

The SELECT statement is used to extract data from one or more tables in the database. Its basic structure is

```
SELECT <columns>
FROM <tables>
WHERE <search conditions>
```

<columns> can be one or more columns from one or more of the tables specified in the following FROM clause.

<tables> is a list of one or more tables which contain the columns listed in the SELECT line.

<search conditions> is a list of conditions that can be used to limit or refine the selection of data that is returned.

Example 1

List all the information available about all the products that are made.

```
SELECT *
FROM PRODUCTS
```

The asterisk (*) in the SELECT statement indicates that all the columns in the tables listed in the following FROM clause should be returned in the order that they are defined in the PRODUCTS table. There is no necessity to specify the WHERE clause, because no conditions for selection are being made.

All the data in the PRODUCTS table is returned.

Example 2

Show the price and name (in that order) of all the products that are made.

```
SELECT P_PRICE, P_NAME
FROM PRODUCTS
```

The result is:	P_PRICE	P_NAME
	13.50	widget
	19.60	fergulator
	15.45	doofer
	8.05	dongle

Example 3

List the name and product number for each of the products that costs more than £10.00.

```
SELECT P_NAME, P_NUMBER
FROM PRODUCTS
WHERE P_PRICE > 10.0
```

The result is:	P_NAME	P_NUMBER
	widget	P1
	doofer	P3
	fergulator	P2

This example shows the use of a simple search condition that results in only rows with a P_PRICE value greater than 10 being returned from the database. Although P_PRICE is used in the query, it does not appear in the result. Often, the unit of value of numerical columns is implicit in the data. We know that the units refer to pounds sterling, because that was the way the database was designed. In an American context, the units would probably refer to U.S. dollars.

The “>” operator is one of the six scalar comparison operators which are used to compare data values:

operator	means
=	equal to
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to
<>	not equal to

GLOSSARY

Comparison operator is used to compare two values.

Search condition is used to limit the data that is returned by a query.

SELECT Statement - Advanced

This section deals briefly with some of the more advanced features of the SELECT statement.

Complex Conditions

Search conditions can be grouped using the *logical operators* AND, OR and NOT.

Example 4

Find the name and price of all the products with a price between £10 and £15.

```
SELECT    P_NAME, P_PRICE
FROM      PRODUCTS
WHERE     P_PRICE >= 10.0 AND P_PRICE <= 15.0
```

The result is:	P_NUMBER	P_NAME	P_PRICE
	P1	widget	13.50

Although the query above is correct, the use of parentheses in the WHERE clause is good practice. This is because an implicit order is applied to the clause when it is processed, and the order in which a clause is processed can affect the result. In some cases, the implicit order may not be what is wanted; parentheses will always over-ride the implicit ordering.

The WHERE clause would better be written:
 WHERE (P_PRICE >= 10.0) AND (P_PRICE <= 15.0)

Precedence Order

There is an implicit order in which logical operations are carried out. This is called the *precedence order*. The order is

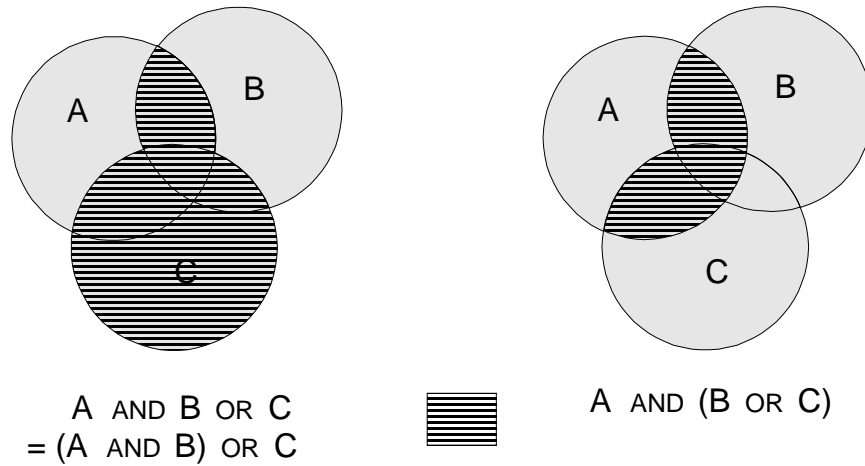
```
NOT (highest)
AND
OR (lowest)
```

For example, the condition

```
a OR b AND c
```

means

```
a OR (b AND c)
```



Where there is a combination of operators, parentheses (some people call these brackets) should be used to group conditions for two reasons.

- To over-ride the order of precedence, you must use parentheses.
- To aid clarity. Even if the logic defined by the precedence rules is what you want, using parentheses makes the logic easy to see.

Example 5

This example uses the ORDERS table, shown below.

ORDERS		
O_NUMBER	CUSTOMER	O_DATE
O1	Smith	29-01-1997
O2	Jones	29-01-1997
O3	Black	03-02-1997
O4	Smith	04-02-1997

Find any orders for the customer named 'Smith' dated the 3rd or 4th of February 1997.

```

SELECT *
FROM ORDERS
WHERE CUSTOMER = 'Smith' AND
(O_DATE = {d '1997-02-04'} OR O_DATE = {d '1997-02-03'})
    
```

Note: At this stage, do not worry about the date format in the WHERE clause; you will learn about this later.

The result is:

O_NUMBER	CUSTOMER	O_DATE
O4	Smith	04-02-1997

Without the use of parentheses in the WHERE clause, that is:

```
WHERE    CUSTOMER = 'Smith' AND
         O_DATE = {d '1997-02-04'} OR O_DATE = {d '1997-02-03' }
```

the returned data would be:

O_NUMBER	CUSTOMER	O_DATE
O3	Black	03-02-1997
O4	Smith	04-02-1997

If the dates had been swapped in the WHERE clause without parentheses, that is:

```
WHERE    CUSTOMER = 'Smith' AND
         O_DATE = {d '1997-02-03'} OR O_DATE = {d '1997-02-04' }
```

then in this case the result would contain order O4 only. Clearly, it is wise to use parentheses to ensure that the query uses the logic you want.

Precedence order also applies to arithmetic operators; again, use parentheses to explicitly state the order of evaluation.

GLOSSARY

Logical operator is used to group conditions.

Precedence order is the implicit order in which operations are carried out

Other Conditional Operators

SQL has other conditional operators, such as BETWEEN, LIKE and NULL. A few examples are presented here.

Example 6

Recall Example 4, “Find the name and price of all the products with a price between £10 and £15.” This could be written using the BETWEEN operator:

```
SELECT    P_NAME, P_PRICE
FROM      PRODUCTS
WHERE     P_PRICE BETWEEN 10.0 AND 15.0
```

This has exactly the same result as using the AND operator in Example 4, but for most people, it is easier to understand.

Example 7

Character data can be queried on the basis of its content.

List all the products which start with the letter “d” and display their price.

```
SELECT    P_NAME, P_PRICE
FROM      PRODUCTS
WHERE     P_NAME LIKE 'd%'
```

When used with the LIKE operator, the % character represents any string of zero or more characters.

In this example, we’ve introduced the character data type. Data types are used to classify data into groups. One of the reasons for doing this is to help prevent errors. If we know that a data item should be numeric, then if a user enters a character, the program can alert the user. In SQL, these are always enclosed in single quote marks, unlike numeric data types. The single quotes are not part of the data itself, so they do not appear in the output. Supported data types are listed in the appendix.

Joining tables

The abstract discussion of SQL showed how tables could be joined. Here we show how this is done using SQL.

Example 8

Generate (partial) data for an invoice. Each invoice for a single order (for example, O4) should contain the order number, product number, product name, quantity and total price.

```
SELECT    O_NUMBER, P_NUMBER, P_NAME, QUANTITY,
TOTAL_PRICE
FROM      PRODUCTS, OLINES
WHERE     (O_NUMBER = 'O4') AND
          (OLINES.P_NUMBER = PRODUCTS.P_NUMBER)
```

The result is:

O_NUMBER	P_NUMBER	P_NAME	QUANTITY	TOTAL_PRICE
O4	P2	fergulator	1	19.60
O4	P4	dongle	1	8.05

Eliminating Duplicates

Recall that two or more rows of a result set may be identical. These duplicate rows can be removed.

Example 9

Which orders contain either fergulators (P2) or dongles (P4), or both

```
SELECT DISTINCT O_NUMBER
FROM           OLINES
WHERE          (P_NUMBER='P2') OR (P_NUMBER='P4')
```

The result is:

O_NUMBER
O1
O3
O4

If the SELECT statement had not contained the DISTINCT keyword, the following data would have been returned:

O_NUMBER
O1
O3
O4
O4

Ordering Data

Although data in tables is not ordered, the result of a query can be ordered using the ORDER BY statement. The default ordering is ascending, but if you want to order the data in descending order, you can, as shown in this example.

Example 10

Find the names and price of all products and arrange them in descending order of price.

```
SELECT      P_NAME, P_PRICE
FROM        PRODUCTS
ORDER BY    P_PRICE DESC
```

The result is:	P_NAME	P_PRICE
	fergulator	19.60
	doofer	15.45
	widget	13.50
	dongle	8.05

DESC in the ORDER BY clause means descending. If you do not use it, then the default order is ascending (if desired, this can be specifically stated using ASC).

Aggregate Functions

There are a number of operations that can be performed, such as finding the average value of data, on a collection of values in one column of a table. These are called *aggregate functions*, and they are listed below.

function	meaning
COUNT DISTINCT	count the number of distinct values in the column
COUNT (*)	count the number of rows in a table
SUM	sum of the values in the column (numeric values only)
AVG	average of the values in the column (numeric values only)
MAX	largest value in the column
MIN	smallest value in the column

Example 11

How many products are there?

```
SELECT COUNT (*)
FROM PRODUCTS
```

The result is:

4

The table that is returned contains a single column with just a single row. The column does not have a name.

Example 12

What is the price of the cheapest product.

```
SELECT MIN (P_PRICE)
FROM PRODUCTS
```

The result is:

8.05

Dealing with dates

When specifying dates with ODBC, we need to use SQL with ODBC extensions. Previously, we stated that the format of dates in a database is application dependent. ODBC must convert this into some standard form. A date is specified in ODBC as follows:

```
{d 'yyyy-mm-dd' }
```

yyyy represents four digits corresponding to the year

mm represents two digits corresponding to the month

dd represents two digits corresponding to the day of the month

For example, 17th January 1995 would be represented as {d '1995-01-17' }

Example 13

Write a query to list all the orders for 4th February 1997.

```
SELECT      *
FROM        ORDERS
WHERE       O_DATE = {d '1997-02-04' }
```

The result is:	O_NUMBER	CUSTOMER	O_DATE
	O4	Smith	04-02-1997

INSERT Statement

The INSERT statement is used to add a row to a table. Its basic structure is

```
INSERT INTO <table> [<column list>]
VALUES <list of values>
```

Example 14

The company has just started to produce a new product, a symmetrical screw, with a price of £0.27. It has been allocated a part number of 6.

```
INSERT INTO PRODUCTS
VALUES ('P6', 'symmetrical screw', 0.27)
```

The PRODUCTS table now contains:

PRODUCTS		
P_NUMBER	P_NAME	P_PRICE
P1	widget	13.50
P2	fergulator	19.60
P3	doofer	15.45
P4	dongle	8.05
P6	symmetrical screw	0.27

In this case, it is not necessary to specify the column names into which data is entered because the order in the VALUES clause matches the column order in the table. However, column names can be specified. The advantage of this is that there is then a partial check that the data is entered in the correct columns. If column names are specified, then the order of the data entered need not match that of the table (but it must match the order specified in the INSERT clause).

```
INSERT INTO PRODUCTS (P_NAME, P_NUMBER, P_PRICE)
VALUES ('symmetrical screw', 'P6', 0.27)
```

The result of this query would be exactly the same as that for the previous query.

UPDATE Statement

The UPDATE statement is used to modify existing rows in a table. Its basic structure is

```
UPDATE    <table>
SET       <column> = <value>
WHERE    <conditions>
```

Every row whose column(s) match the conditions is updated.

Example 15

Say that widgets have been reduced in price to £12.20. Update the PRODUCTS table to reflect this.

```
UPDATE    PRODUCTS
SET       P_PRICE = 12.20
WHERE    P_NAME = 'widget'
```

This table now contains

PRODUCTS		
P_NUMBER	P_NAME	P_PRICE
P1	widget	12.20
P2	fergulator	19.60
P3	doofer	15.45
P4	dongle	8.05
P6	symmetrical screw	0.27

Example 16

Say that all product prices are raised by 10%. Updating the table could be done like this.

```
UPDATE    PRODUCTS
SET       P_PRICE = P_PRICE * 0.1
```

This table now contains

PRODUCTS		
P_NUMBER	P_NAME	P_PRICE
P1	widget	13.42
P2	fergulator	8.86
P3	doofer	17.00
P4	dongle	21.56
P6	symmetrical screw	0.30

This result shows the effects of rounding. In our example, we rounded up. Exactly what happens is application dependent.

DELETE Statement

The DELETE statement is used to delete one or more rows from a table. Its basic structure is

```
DELETE
FROM   <table>
WHERE  <conditions>
```

Example 17

Say that doofers are no longer made - delete that product for the PRODUCTS table

```
DELETE
FROM   PRODUCTS
WHERE  P_NAME = 'doofer'
```

An alternative WHERE clause is: WHERE P_NUMBER=P3. This has exactly the same effect, since there is a 1:1 correspondence between P_NUMBER and P_NAME. The result in both cases is:

PRODUCTS		
P_NUMBER	P_NAME	P_PRICE
P1	widget	13.42
P2	fergulator	8.86
P4	dongle	21.56
P6	symmetrical screw	0.30

Just a single row has been deleted. The next example shows the deletion of many rows.

Example 18

Say that order number O4 had been entered by mistake. Delete this order from the OLINES table (in practice, an application running the SQL would probably have a method for dealing with incorrect entries. It's not a good idea to allow users to delete data at will).

```
DELETE
FROM   OLINES
WHERE  O_NUMBER = 4
```

This table now contains

OLINES			
O_NUMBER	P_NUMBER	QUANTITY	TOTAL_PRICE
O1	P1	1	13.50
O1	P4	2	16.10
O3	P4	1	8.05

Omitting the WHERE clause would have resulted in all the rows being deleted.

Summary

In this module you have learnt that

- ★ SQL (Structured Query Language) is a language used for relational databases
- ★ data is perceived to be stored in tables
- ★ tables are composed of rows and columns
- ★ indexes can be used to speed up data retrieval
- ★ an index can be unique or non-unique
- ★ the basic data manipulation statements are: SELECT, UPDATE, INSERT, DELETE
- ★ data can be combined from different tables using joins
 - Inner joins select records that have matching values in both tables
 - Outer joins select all records from one table; the other table may not have matching values
- ★ NULL represents a case where a value is unknown
- ★ there are six scalar comparison operators (<, >, <>, <=, >=, =) which are used to compare data values
- ★ there are three logical operators (AND, OR, NOT) which are used to combine conditions
- ★ parentheses
 - over-ride precedence order
 - allow you to see clearly the logic of a condition
- ★ aggregate functions are group operations, such as MAX, AVG, which work on collections of data values

SQL Review Questions

1. What type of databases is SQL designed for?
2. What is the purpose of a join?
3. What is the difference between an inner join and an outer join?
4. What do the terms 'left' and 'right' mean in the context of outer joins?
5. What three operators are used to combine conditions in SQL?
6. Is it always necessary to use parentheses when using an AND or OR operator in SQL?
7. The result of an SQL query may contain duplicate values. How can these duplicates be removed?
8. How many comparison operators are there? List them.

SQL Review Answers

1. What type of databases is SQL designed for?

Relational databases.

2. What is the purpose of a join?

To obtain information from more than one table

3. What is the difference between an inner join and an outer join?

An inner join returns data for the join column only if it occurs in both tables
An outer join returns data from the join column whether or not it appears in both tables.

4. What do the terms 'left' and 'right' mean in the context of outer joins?

They refer to the relative position of the table in the join condition. The first table is the left table, and the second table is the right table.

5. What three operators are used to combine conditions in SQL?

AND, OR, NOT.

6. Is it always necessary to use parentheses when using an AND or OR operator in SQL?

No, but to ensure that the result is based on the condition you thought you had defined, and for clarity, it's a good idea always to use them.

7. The result of an SQL query may contain duplicate values. How can these duplicates be removed?

Use SELECT DISTINCT.

8. How many scalar comparison operators are there? List them.

There are 6:

= (equal), < (less than), > (greater than), <= (less than or equal),
>= (greater than or equal), <> (not equal).

Further Exercises

For your convenience when answering the questions which require the writing of SQL code, the three tables are replicated here.

PRODUCTS		
P_NUMBER	P_NAME	P_PRICE
P1	widget	13.50
P2	fergulator	19.60
P3	doofer	15.45
P4	dongle	8.05

ORDERS		
O_NUMBER	CUSTOMER	O_DATE
O1	Smith	29-01-1997
O2	Jones	29-01-1997
O3	Black	03-02-1997
O4	Smith	04-02-1997

OLINES			
O_NUMBER	P_NUMBER	QUANTITY	TOTAL_PRICE
O1	P1	1	13.50
O1	P4	2	16.10
O3	P4	1	8.05
O4	P2	1	19.60
O4	P4	1	8.05

1. Write a query to show all the orders.
2. Write a query to show all the order numbers and the customer who sent the order.
3. Write a query to list all the orders for 1st January 1997 - show the order number and the customer only.

4. Using SQL, what is the easiest way to find out how many orders there are?

5. Use SQL to calculate the total value of all the orders.

6. Find the average price of all the products by writing a simple query.

7. Write a query to list the name and product number of all the products despatched on 29th January 1997.

8. Write a query to list all the customers in alphabetic order.

9. Write a query to show the name of all products and their price; list this information in descending price order.

Suggested Answers to Exercises

1. Write a query to show all the orders.

```
SELECT    *
FROM      ORDERS
```

2. Write a query to show all the order numbers and the customer who sent the order.

```
SELECT    O_NUMBER, CUSTOMER
FROM      ORDERS
```

3. Write a query to list all the orders for 1st January 1997 - show the order number and the customer only.

```
SELECT    O_NUMBER, CUSTOMER
FROM      ORDERS
WHERE     O_DATE = {d '1997-01-01' }
```

Here, we must use an ODBC extension to SQL to specify the date.

4. Using SQL, what is the easiest way to find out how many orders there are?

```
SELECT    COUNT (*)
FROM      ORDERS
```

5. Use SQL to calculate the total value of all the orders.

```
SELECT    SUM (TOTAL_PRICE)
FROM      OLINES
```

6. Find the average price of all the products by writing a simple query.

```
SELECT    AVG (P_PRICE)
FROM      PRODUCTS
```

7. Write a query to list the name and product number of all the products despatched on 29th January 1997.

```
SELECT    P_NAME, P_NUMBER
FROM      OLINES, PRODUCTS
WHERE     O_DATE = {d '1997-01-29'} AND
          OLINES.P_NUMBER = PRODUCTS.P_NUMBER
```

Here, we must use an ODBC extension to SQL to specify the date.

8. Write a query to list all the customers in alphabetic order.

```
SELECT DISTINCT CUSTOMER
FROM      ORDERS
ORDER BY  CUSTOMER
```

Note: in practice, this query would be posed against a table dedicated to customer information.

9. Write a query to show the name of all products and their price; list this information in descending price order.

```
SELECT    P_NAME, P_PRICE
FROM      PRODUCTS
ORDER BY  P_PRICE DESC
```

Additional Notes

The Easysoft Catalog

The Easysoft Catalog can be considered as a set of tables which themselves contain the information needed to describe the non-relational server data in relational terms, so that it may be viewed by ODBC-compliant applications (which interact with data using SQL). These tables can be queried by SQL just as any other table can.

Supported SQL

The following SQL statements are supported by the Easysoft software. Words enclosed in angle brackets (<>) should be replaced by any valid SQL syntax.

INSERT

```
INSERT INTO <table> [( <column identifiers>)] VALUES (<values>)
```

SELECT

```
SELECT [ALL | DISTINCT] <select-list>  
FROM <table>  
[WHERE <search condition>]  
[GROUP BY <column names>]  
[HAVING <search condition>]  
[ORDER BY <sort specification>]
```

DELETE SEARCHED

```
DELETE FROM <table> [WHERE <search condition>]
```

UPDATE SEARCHED

```
UPDATE <table>  
SET <column identifiers> = <expression>  
[WHERE <search condition>]
```

Supported Set (Aggregate) Functions

COUNT(*), COUNT, MAX, MIN, SUM, AVG

Reserved Words

SQL contains reserved words (a word that is part of the language itself), which should not be used when writing SQL commands unless they are expressed as character literals (i.e. enclosed between single quote marks). Examples of reserved words are SELECT, UPDATE, BETWEEN and SET. These words are defined by the standards organisations that define the SQL language, and are listed below.

ABSOLUTE	COLLATION_NAME	DICTIONARY
ACTION	COLLATION_SCHEMA	DISCONNECT
ADA	COLUMN	DISTINCT
ADD	COLUMN_NAME	DOMAIN
AFTER	COMMAND_FUNCTION	DOUBLE
ALIAS	COMMIT	DROP
ALL	COMMITTED	DYNAMIC_FUNCTION
ALLOCATE	COMPLETION	EACH
ALTER	CONDITION_NUMBER	ELSE
AND	CONNECT	ELSEIF
ANY	CONNECTION	END
ARE	CONNECTION_NAME	END-EXEC
AS	CONSTRAINT	EQUALS
ASC	CONSTRAINT_CATALOG	ESCAPE
ASSERTION	CONSTRAINT_NAME	EXCEPT
ASYN	CONSTRAINT_SCHEMA	EXCEPTION
AT	CONSTRAINTS	EXEC
AUTHORIZATION	CONTINUE	EXECUTE
AVG	CONVERT	EXISTS
BEFORE	CORRESPONDING	EXTERNAL
BEGIN	COUNT	EXTRACT
BETWEEN	CREATE	FALSE
BIT	CROSS	FETCH
BIT_LENGTH	CURRENT	FIRST
BOOLEAN	CURRENT_DATE	FLOAT
BOTH	CURRENT_TIME	FOR
BREADTH	CURRENT_TIMESTAMP	FOREIGN
BY	CURRENT_USER	FORTRAN
C	CURSOR	FOUND
CALL	CURSOR_NAME	FROM
CASCADE	CYCLE	FULL
CASCADE	DATA	GENERAL
CASE	DATE	GET
CAST	DATETIME_INTERVAL_CODE	GLOBAL
CATALOG	DATETIME_INTERVAL_PRECISIO	GO
CATALOG_NAME	N	GOTO
CHAR	DAY	GRANT
CHAR_LENGTH	DEALLOCATE	GROUP
CHARACTER	DEC	HAVING
CHARACTER_LENGTH	DECIMAL	HO
CHARACTER_SET_NAME	DECLARE	IDENTITY
CHARACTER_SET_SCHEMA	DEFAULT	IF
CHECK	DEFERRABLE	IGNORE
CLASS_ORIGIN	DEFERRED	IMMEDIATE
CLOSE	DELETE	IN
COALESCE	DEPTH	INCLUDE
COBOL	DESC	INDEX
COLLATE	DESCRIBE	INDICATOR
COLLATION	DESCRIPTOR	INITIALLY
COLLATION_CATALOG	DIAGNOSTICS	INNER

INPUT	ORDER	SMALLINT
INSENSITIVE	OTHERS	SOME
INSERT	OUTER	SPACE
INT	OUTPUT	SQL
INTEGER	OVERLAPS	SQLCA
INTERSECT	PAD	SQLCODE
INTERVAL	PARAMETERS	SQLERROR
INTO	PARTIAL	SQLEXCEPTION
IS	PASCAL	SQLSTATE
ISOLATION	PENDANT	SQLWARNING
JOIN	PLI	STRUCTURE
KEY	POSITION	SUBCLASS_ORIGIN
LANGUAGE	PRECISION	SUBSTRING
LAST	PREORDER	SUM
LEADING	PREPARE	SYSTEM
LEAVE	PRESERVE	SYSTEM_USER
LEFT	PRIMARY	TABLE
LENGTH	PRIOR	TABLE_NAME
LESS	PRIVATE	TEMPORARY
LEVEL	PRIVILEGES	TEST
LIKE	PROCEDURE	THEN
LIMIT	PROTECTED	THERE
LOCAL	PUBLIC	TIME
LOOP	READ	TIMESTAMP
LOWER	REAL	TIMEZONE_HOUR
MATCH	RECURSIVE	TIMEZONE_MINUTE
MAX	REF	TO
MESSAGE_LENGTH	REFERENCES	TRACEPOINT
MESSAGE_OCTET_LENGTH	REFERENCING	TRAILING
MESSAGE_TEXT	RELATIVE	TRANSACTION
MIN	REPEATABLE	TRANSLATE
MINUTE	REPLACE	TRANSLATION
MODIFY	RESIGNAL	TRIGGER
MODULE	RESTRICT	TRIM
MONTH	RETURN	TRUE
MORE	RETURNED_LENGTH	TYPE
MUMPS	RETURNED_OCTET_LENGTH	UNCOMMITTED
NAME	RETURNED_SQLSTATE	UNDER
NAMES	RETURNS	UNION
NATIONAL	REVOKE	UNIQUE
NATURAL	RIGHT	UNKNOWN
NCHAR	ROLE	UNNAMED
NEW	ROLLBACK	UPDATE
NEXT	ROUTINE	UPPER
NO	ROW	USAGE
NONE	ROW_COUNT	USER
NOT	ROWS	USING
NOTRACEPOINT	SAVEPOINT	VALUE
NULL	SCALE	VALUES
NULLABLE	SCHEMA	VARCHAR
NULLIF	SCHEMA_NAME	VARIABLE
NUMBER	SCROLL	VARYING
NUMERIC	SEARCH	VIEW
OBJECT	SECOND	VIRTUAL
OCTET_LENGTH	SECTION	VISIBLE
OF	SELECT	WAIT
OFF	SENSITIVE	WHEN
OID	SEQUENCE	WHENEVER
OLD	SERIALIZABLE	WHERE
ON	SERVER_NAME	WHILE
ONLY	SESSION	WITH
OPEN	SESSION_USER	WITHOUT
OPERATION	SET	WORK
OPERATORS	SIGNAL	WRITE
OPTION	SIMILAR	YEAR
OR	SIZE	ZONE

Bibliography



This bibliography lists a few sources of information relevant to information in this module. Entries are listed alphabetically by title, except that initial articles (i.e. 'A' and 'The') are ignored for the purposes of ordering.

Database Language SQL, Document ANSI X3.135-1986, American National Standards Institute, 1986.

This is the original SQL standard definition. It is a heavy-duty reference.

Database Language SQL with Integrity Enhancement, Document ISO/IEC 9075: 1989(E), International Organisation for Standardisation, 1989.

This document defines additions to the original SQL standard which deal with the integrity of data. It is a heavy-duty reference.

Data Management: Structured Query Language (SQL), Version 2. X/Open CAE Specification. X/Open Document Number:C449. IISBN: 1-85912-151-9.

Database Systems Engineering, R. P. Whittington, Clarendon Press, Oxford, 1988. ISBN 0-19-859666-9

Fundamentals of Database Systems, R. Elmasri & S. B. Navathe, Benjamin/Cummings Publishing Co. Inc., 1989. ISBN 0-201053090-2

An Introduction to Database Systems, volume 1, C. J. Date, Addison-Wesley Publishing Company, Inc., 1990. ISBN 0-201-52878-9

One of the most popular books on databases and SQL (and it covers more than just relational systems). Although it calls itself an introduction, at nearly 1000 pages it's not entirely a trivial read. Nevertheless, it starts simply, and contains numerous examples and exercises (and answers).

Introduction to SQL, R. F. van der Lans, Addison-Wesley Publishing Company, Inc., 1988. ISBN 0-201-17521-5

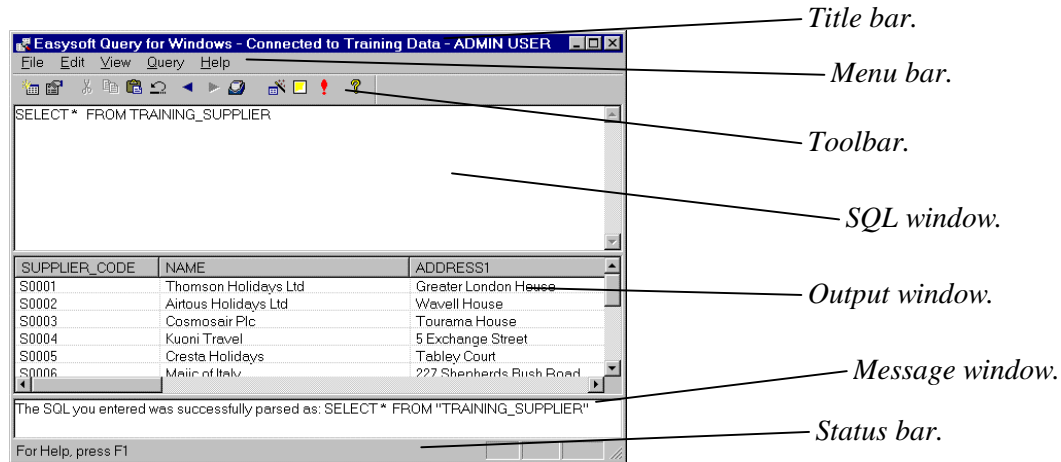
Rather an old book, so some of the material is dated. However, the principles remain sound. Packed with worked examples, exercises (and answers).

A Relational Model of Data for Large Shared Data Banks, E. F. Codd, Communications of the ACM, volume 13, number 6 (June 1970).

A technical paper, not suitable for the beginner.

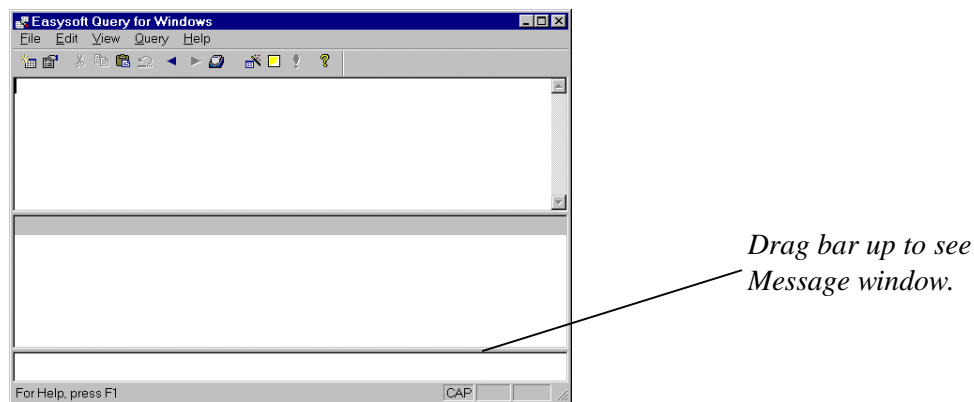
Easysoft Query for Windows

The figure below shows the Easysoft Query for Windows main dialog box.

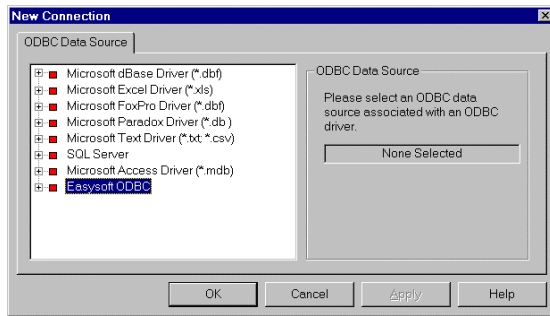


Start and Connect

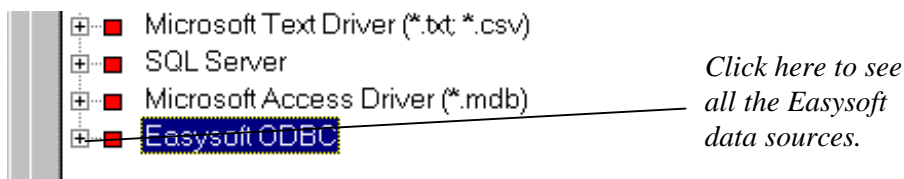
1. Start Easysoft Query for Windows by clicking on the shortcut that has been created:



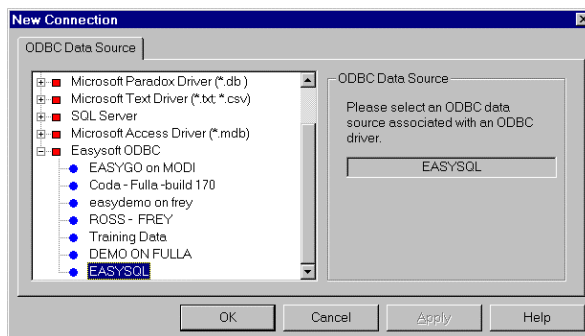
2. Connect to a data source by selecting the **File, New Connection** menu sequence. The New Connection dialog box appears.



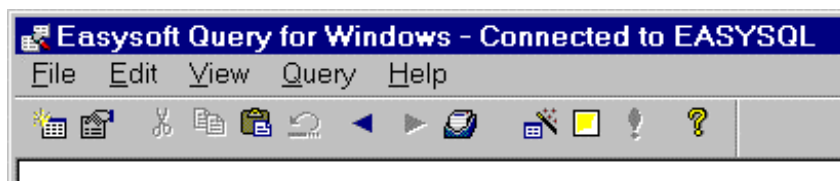
- The data sources are organised by driver type. To see all the data sources that use the Easysoft driver, click on the plus sign at the left of Easysoft ODBC.



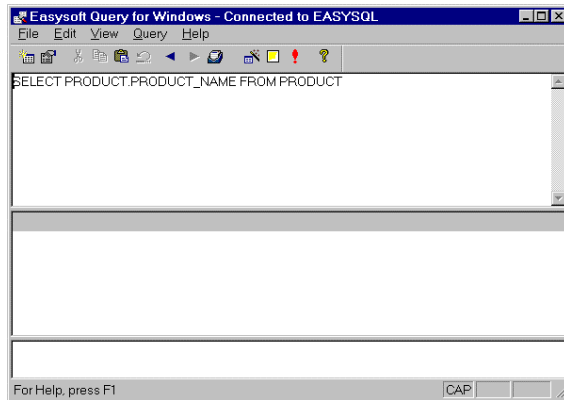
- A list of available data sources is displayed. Highlight EASYSQL and click **OK**.




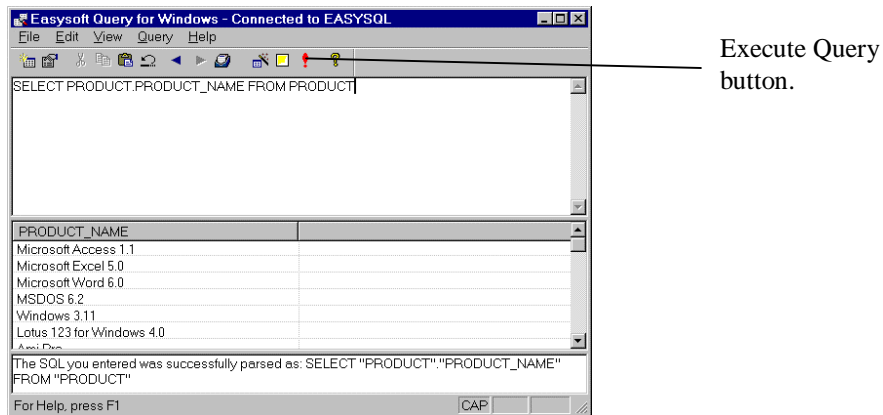
- The title bar now shows which data source you are connected to.



6. Now that you have connected to a data source, you can send SQL to it. Enter this test query in the SQL window.



7.  Click the Execute Query button (or select **Query, Execute** from the menu options or press F5) to run the query. The query is cleared from the window in preparation for the following query, and the result appears in the output window.

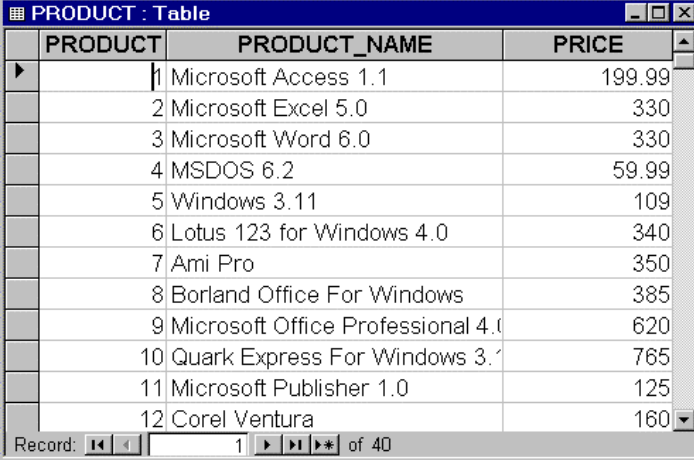


8. To close Easysoft Query for Windows select **Exit** from the **File** menu. (Do not close Easysoft Query for Windows at this stage).

The Example Database

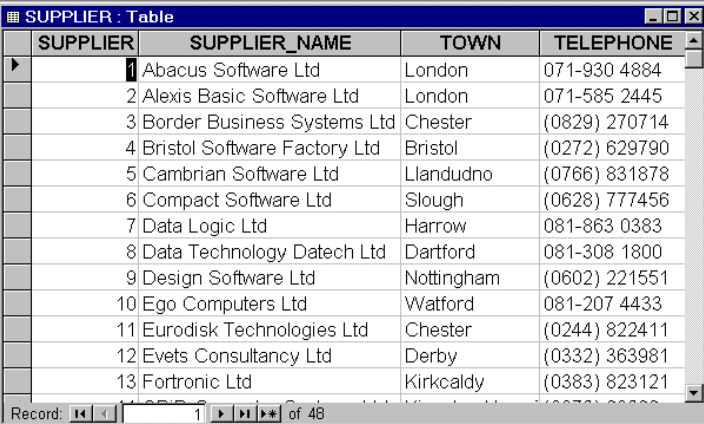
There are three tables in this database, PRODUCT, SUPPLIER and SUPPLIER_PRODUCT. The columns in the tables, and some of the data for each of the tables is shown in the screen shots below.

PRODUCT table



PRODUCT	PRODUCT_NAME	PRICE
1	Microsoft Access 1.1	199.99
2	Microsoft Excel 5.0	330
3	Microsoft Word 6.0	330
4	MSDOS 6.2	59.99
5	Windows 3.11	109
6	Lotus 123 for Windows 4.0	340
7	Ami Pro	350
8	Borland Office For Windows	385
9	Microsoft Office Professional 4.0	620
10	Quark Express For Windows 3.1	765
11	Microsoft Publisher 1.0	125
12	Corel Ventura	160

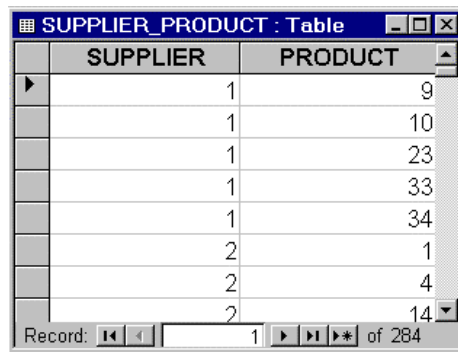
SUPPLIER table



SUPPLIER	SUPPLIER_NAME	TOWN	TELEPHONE
1	Abacus Software Ltd	London	071-930 4884
2	Alexis Basic Software Ltd	London	071-585 2445
3	Border Business Systems Ltd	Chester	(0829) 270714
4	Bristol Software Factory Ltd	Bristol	(0272) 629790
5	Cambrian Software Ltd	Llandudno	(0766) 831878
6	Compact Software Ltd	Slough	(0628) 777456
7	Data Logic Ltd	Harrow	081-863 0383
8	Data Technology Datech Ltd	Dartford	081-308 1800
9	Design Software Ltd	Nottingham	(0602) 221551
10	Ego Computers Ltd	Watford	081-207 4433
11	Eurodisk Technologies Ltd	Chester	(0244) 822411
12	Evets Consultancy Ltd	Derby	(0332) 363981
13	Fortronic Ltd	Kirkcaldy	(0383) 823121

SUPPLIER_PRODUCT table

This table links Products to Suppliers.



SUPPLIER	PRODUCT
1	9
1	10
1	23
1	33
1	34
2	1
2	4
2	14

Exercises

Write the SQL in the space below the questions, then use Easysoft Query for Windows to send the query to the server. The database is described in the previous section.

1. Write a query to show all the products.
2. Write a query to show all the suppliers and their telephone numbers.
3. List all products costing more than £300.00. Also include the price of the product.
4. How many products are there?
5. How many products cost over £300.00?
6. List all Microsoft products.

Suggested Answers

1. Write a query to show all the products.

```
SELECT *  
FROM PRODUCT
```

2. Write a query to show all the suppliers and their telephone numbers.

```
SELECT SUPPLIER_NAME, TELEPHONE  
FROM SUPPLIER
```

3. List all products costing more than £300.00. Also include the price of the product.

```
SELECT PRODUCT_NAME, PRICE  
FROM PRODUCT  
WHERE PRICE > 300
```

4. How many products are there?

```
SELECT COUNT(*)  
FROM PRODUCT
```

5. How many products cost over £300.00?

```
SELECT COUNT(PRODUCT_NAME)  
FROM PRODUCT  
WHERE PRICE > 300
```

6. List all Microsoft products.

```
SELECT PRODUCT_NAME  
FROM PRODUCT  
WHERE PRODUCT_NAME like 'Microsoft%'
```

7. For each supplier, list the products that are supplied.

```
SELECT SUPPLIER_NAME, PRODUCT_NAME
FROM SUPPLIER, SUPPLIER_PRODUCT, PRODUCT
WHERE SUPPLIER.SUPPLIER = SUPPLIER_PRODUCT.SUPPLIER AND
SUPPLIER_PRODUCT.PRODUCT = PRODUCT.PRODUCT
```

8. List all the products supplied by the supplier named "Compact Software Ltd". Include the supplier's name in the result.

```
SELECT SUPPLIER_NAME, PRODUCT_NAME
FROM SUPPLIER, SUPPLIER_PRODUCT, PRODUCT
WHERE SUPPLIER.SUPPLIER_NAME = 'Compact Software Ltd' AND
SUPPLIER.SUPPLIER = SUPPLIER_PRODUCT.SUPPLIER AND
SUPPLIER_PRODUCT.PRODUCT = PRODUCT.PRODUCT
```

9. List all suppliers of Microsoft Word. (Hint: use PRODUCT table first)

```
SELECT PRODUCT_NAME, SUPPLIER_NAME
FROM PRODUCT, SUPPLIER_PRODUCT, SUPPLIER
WHERE PRODUCT.PRODUCT_NAME like 'Microsoft Word%' AND
SUPPLIER_PRODUCT.PRODUCT = PRODUCT.PRODUCT AND
SUPPLIER.SUPPLIER = SUPPLIER_PRODUCT.SUPPLIER
```

3. Introduction to RMS Files

In this module you will learn about

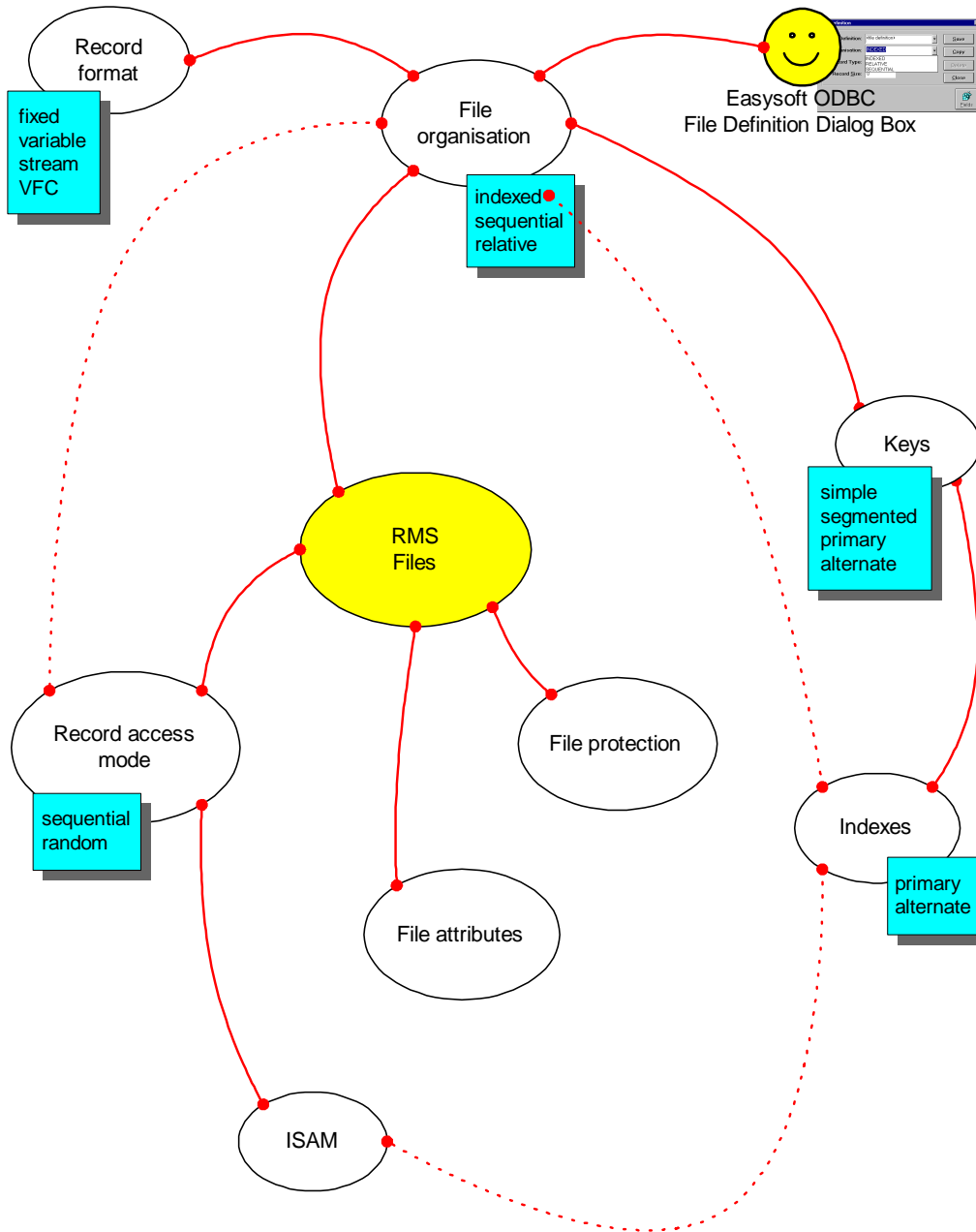
- ★ Fixed and variable record formats
- ★ The different ways of accessing files
- ★ Indexed, relative and sequential file organisations
- ★ File attributes, for example, organisation and record format
- ★ File protection

At the end of the module there are some review questions.

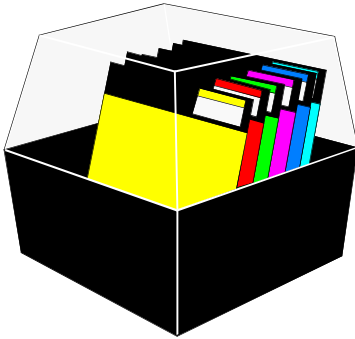
Contents

File Organisation Overview	3-3
File Attributes	3-5
Sequential Files	3-6
Relative Files	3-7
Indexed Files	3-9
Keys	3-9
Indexes	3-12
Record Access Mode	3-15
Record Format	3-17
File Protection	3-20
UIC-Based Protection	3-20
ACL-Based Protection	3-22
Summary	3-23
RMS Review Questions	3-24
CURRENCY_RATE.DAT	3-25
RMS Review Answers	3-26
Exercise Model Answers	3-28
Bibliography	3-30

RMS Mind Map



File Organisation Overview



A file is a collection of related information whose requirements are established by the nature of the application programs needing the information. For example, a company might maintain product information in one file and information about employees in another file.

Within a file, the information is divided into records. For example, each record in a personnel file would contain information about a single employee. In turn, each record is further divided into discrete units of information known as fields. The user who creates the file defines the number, location within the record and the logical interpretation of the fields.

RMS (Record Management Services) is one of the data management facilities provided by the OpenVMS operating system. It interprets the record structure of files and it provides a set of routines which is used to manipulate files.

RMS provides three basic *file organisations*, that is, methods of organising the records in a file, namely Sequential, Relative and Indexed. Related to, but different from, the file organisation is the *access mode*, that is, the way in which records are stored and retrieved. Another related topic is *record format*, that is, the way the record physically appears on the recording medium.

The remainder of this section looks at the advantages and disadvantages of the three file organisations. The organisations are then described in more detail. The last two sections deal with record access modes and record formats.

GLOSSARY

Field A segment of a data record.

File organisation (in VAX/OpenVMS terms) The file structure that is used as the physical arrangement of the data on the storage medium. *RMS* file organisations are sequential, indexed and relative.

Record A collection of related data items treated as a unit. A record contains one or more fields.

Record access mode The method used in *RMS* for retrieving and storing records in a file.

Record format The physical structure of a record on the storage medium.

RMS (Record Management Services) One of the data management facilities provided by the OpenVMS operating system.

The table below outlines the advantages and disadvantages of the three file organisations.

File Organisations - Advantages and Disadvantages	
Advantages	Disadvantages
Sequential	
<ul style="list-style-type: none"> • Efficient use of disk and memory. • Optimal usage if the application accesses all records sequentially on each run. • Most flexible record format; data can be exchanged with systems other than RMS. • Allows easy file extension. 	<ul style="list-style-type: none"> • Some high-level languages allow sequential access only. • Records can only be added to end of file. • Allows write access by multiple concurrent users only in limited cases.
Relative	
<ul style="list-style-type: none"> • Allows both sequential and random access for all languages. • Provides random record deletion and insertion. • Allows records to be read- and write-shared. 	<ul style="list-style-type: none"> • Records can only be stored on disk. • Requires files to contain a record cell for each relative record number allocated - files may not be densely populated. • Record cells must all be the same size.
Indexed	
<ul style="list-style-type: none"> • Allows sequential and random access by key value of all languages. • Allows random record deletion and insertion. • Records can be read- and write-shared. • Variable length records can change size on update. • Easy file extension. 	<ul style="list-style-type: none"> • Can only store data on disk. • Requires more disk space. • Uses more processing power to process records. • In general, requires multiple disk accesses to randomly process a record.

Note

Do not confuse the terms *file organisation* and *file type*. File organisation refers to the structure of the records in the file. File type refers to the naming of a file; a similar

concept in other environments is known as the file extension.

File Attributes

When an RMS file is created, its attributes, that is, its physical characteristics, must be defined. These attributes are defined by source language statements in an application program or by an RMS utility. The following are assigned (you will see examples in later modules):

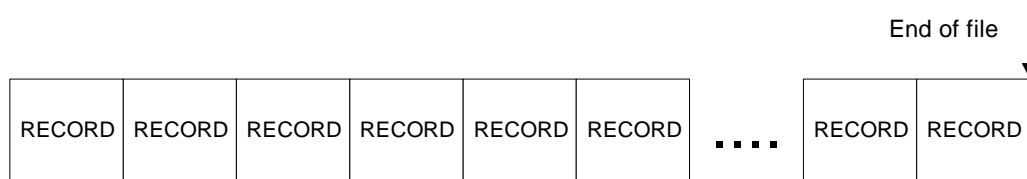
- file name, file size and file location
- file organisation
- the owner's user identification code (UIC)
- a protection code
- device (depends upon the organisation of the file)
- record format and size
- keys (indexed files only)

When an RMS file is created on a disk volume, the user can specify an initial allocation size. If this is not done, then RMS itself allocates the minimum amount of storage needed to contain the defined attributes of the file. The initial size can be increased later. The starting location of the file can be specified optionally using a volume-relative block number or physical address.

Sequential Files

In a sequential file organisation, the records appear in consecutive sequence. The order in which records appear is always the order in which they were originally written to the file. This is shown below.

Records in a sequential file



The records can be fixed length or variable length (see “Record Format”, page 3-17). There are no empty records between records that contain data, and this limits the operations on the file to:

- Reading data from any record
- Writing data by adding records at the end of the file

Records cannot be deleted once they have been added to the file.

Records can be updated, but the record length cannot be increased.

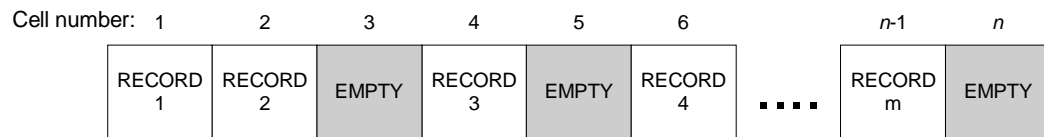
This is the only organisation which can be used for nondisk devices such as magnetic tape. It is the simplest organisation, and also it is optimal if the application assess all the records on each run.

Relative Files

A relative file consists of a series of fixed-length record positions (or cells) which are numbered consecutively from 1 to n . This enables RMS to calculate the physical position of a record on the disk. The number, referred to as the *relative record number*, indicates the record cell's position relative to the start of the file.

Each record in the file may be randomly assigned to a specific cell. For example, the first and second records may be assigned to the first and second cells and the third record may be assigned to the fourth cell, leaving the third cell empty. Unused cells and cells from which records have been deleted may be used to store new records. This organisation is shown below.

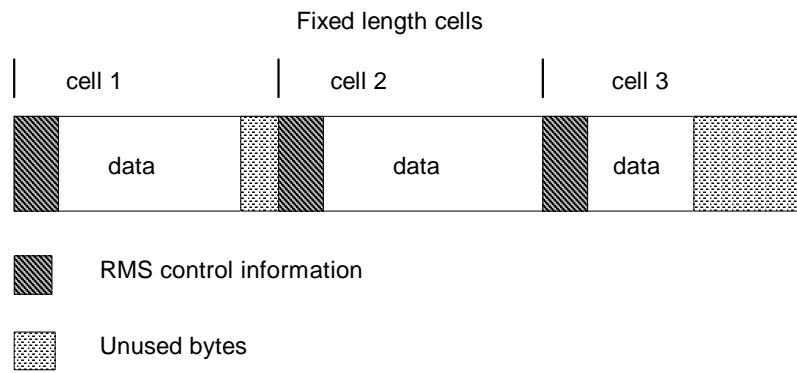
Records in a relative file



In a relative file, the actual length of the individual records may vary (that is, different size records can be in the same file) up to the limits imposed by the specified cell length. Record format is discussed in detail on page 3-17.

Note - Variable length records in a relative file

After reading about record formats on page 3-17, you may wonder how variable length records are stored in relative files, since a relative file consists of a series of fixed-length cells. Here's how it's done.



In this case, the control information is contained in a 3 byte field. The first portion of this is the 2-byte count field. Then there is a 1 byte field which indicates whether or not the cell is empty (a delete flag). These bytes are used only by RMS - you do not need to be concerned with them, except when planning the file's space requirements.

Indexed Files

An indexed file consists of data records and an index structure which is used to access those records. The keys in the record are used by the index. First we look at keys in general, and then at indexes.

Keys

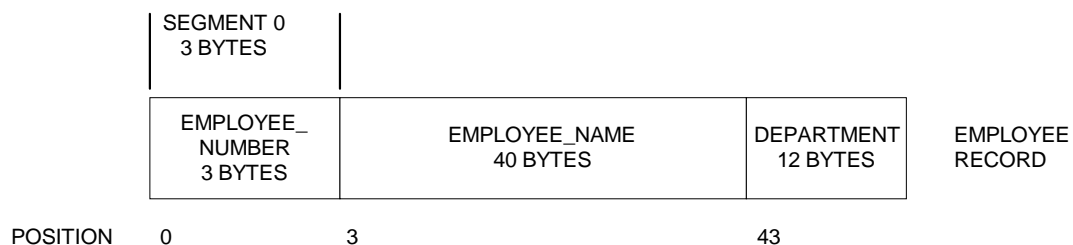
In an indexed file (described in "Indexes", page 3-12), each record includes one or more key fields (or keys). Each key is identified by its location, its length, and whether it is a simple or a segmented key.

A simple key is a key that has just one section (segment) which can be any one of the following data types:

- A single contiguous character string
- A packed decimal number
- A 2-, 4-, or 8-byte unsigned binary number
- A 2-, 4-, or 8-byte signed integer

The figure below shows a simple key which corresponds to employee number.

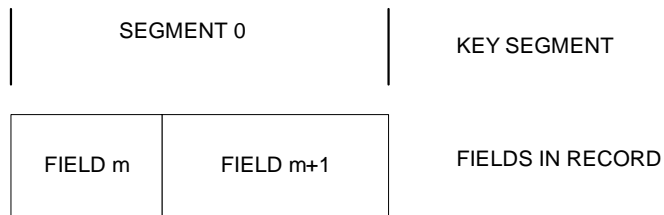
A simple key



Segmented Keys

A key can be defined on more than one field in a record. If the fields are contiguous (i.e. there are no intervening fields), then a key with a single segment can be defined over these fields. This is shown below.

A simple key defined on two fields



This is an acceptable structure, since the segment is a contiguous set of bytes.

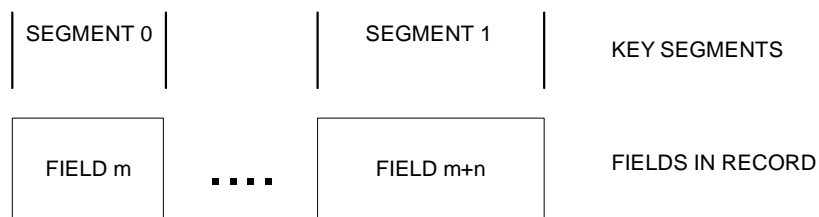
For example, say that a file, EMPLOYEE.DAT, contains information about employees. Imagine that the fields were defined to be in the following order:

EMPLOYEE_NUMBER, DEPARTMENT, EMPLOYEE_NAME

In this case, a simple key could be defined over the EMPLOYEE_NUMBER and DEPARTMENT fields.

However, if we wanted to define a single key over say two fields that were not adjacent, then we would need two segments in the key, as shown below.

A segmented key defined on two fields



For example, reconsider the EMPLOYEE.DAT file. Imagine that the fields were defined to be in the following order:

EMPLOYEE_NUMBER, EMPLOYEE_NAME, DEPARTMENT

In this case, it is not possible to define a simple key over the EMPLOYEE_NUMBER and DEPARTMENT fields, because they are not contiguous. You could still define a key over these fields, but it would be a segmented key consisting of two segments.

Segmented keys enhance flexibility in manipulating data files by letting you select the placement of data fields and then tailoring the key structure to fit this layout. You can improve performance by defining a segment that contains the desired key together with another segment that contains a unique field, thereby making the entire key unique. When only non-contiguous portions of a text string are needed for a key, you can improve efficiency by defining smaller keys that include only these segments.

Notes

1. Do not confuse the terms *simple key* and *single key*. *Simple key* is a technical term referring to a key which has only one segment. In the term *single key*, the word *single* is used in its everyday sense to mean one (1). One (single) key may be simple or segmented.
2. For segments which use the character string data type, the segment has a maximum length of 255 bytes.
3. A segmented key can be composed of a maximum of 8 segments.

Primary and Alternate Keys

Within the context of RMS indexed files, the *primary key* is the mandatory key within the data records of an indexed file. It is used to determine the placement of the records within the file and to build the primary index (described in the next section).

An *alternate key* is an optional key within the data records in an indexed file. It is used to build an alternate index.

For example, an employee record could be defined as having a primary key on the EMPLOYEE_NUMBER field, and an alternate key on the DEPARTMENT field.

Primary and alternate keys

Primary key		Alternate key	
EMPLOYEE_NUMBER	EMPLOYEE_NAME	DEPARTMENT	EMPLOYEE RECORD

GLOSSARY

Alternate key (in RMS terms) An optional key in an indexed file.

Contiguous A technical term meaning adjacent, next to or continuous.

Key One or more characters used to identify or find a record.

Primary key (in RMS terms) The mandatory key within the records of an indexed file.

Relative record number An identification number that specifies the position of a record cell relative to the beginning of the file.

Segmented key A key that consists of separate segments (sections) in different parts of a record.

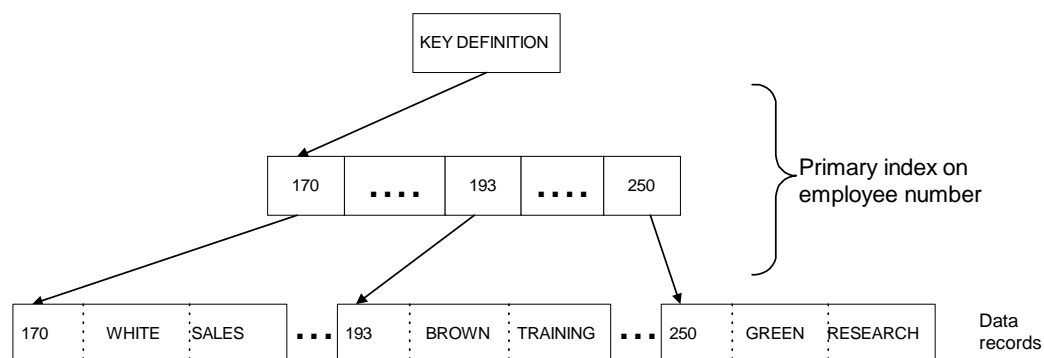
Simple key A key that consists of a single segment (section) of a record.

Indexes

The indexed file organisation lets you store data records in an index structure which is ordered by the primary key and it lets you retrieve data using index structures ordered by primary or alternate keys.

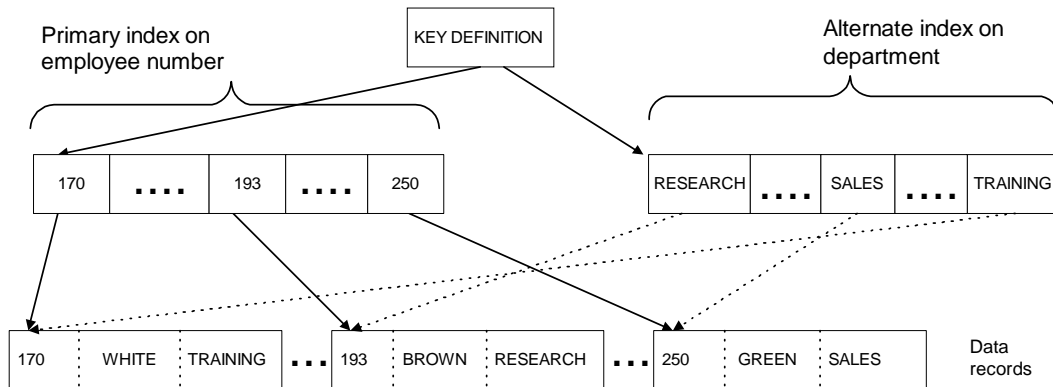
For example, an indexed file containing records with fields corresponding to employee number, employee name and department may be ordered in ascending sort order by the primary key - employee number. This is shown below.

Single key indexed file organisation



However, you may want to set up *alternate indexes* (i.e. additional indexes) for retrieving records from the file. For example, you could set up an alternate index on department, as shown below.

Multiple key indexed file organisation



In addition to the indexes, each indexed file includes a *prolog* structure that contains information about the file, including file attributes. RMS currently supports three distinct prologs - Prolog 1, Prolog 2 and Prolog 3. Normally RMS creates a Prolog 3 indexed file, but you can specify a previous prolog version, typically for compatibility with RMS-11.

The physical location of records in an indexed file is transparent to your program because RMS controls record placement.

GLOSSARY

Alternate index An index which is additional to the primary index.

Index The structure that allows retrieval of records in an indexed file by key value.

Note - Key and Index

Do not confuse the terms *key* and *index*. *Key* refers to a section of a file - a field. *Index* refers to the structure that is used to access the data. This distinction is exemplified by the following quote, taken from the *Guide to OpenVMS File Applications*, chapter 2.

“For an indexed file, you must define at least one key, the primary key, and you can optionally define one or more alternate keys. RMS uses alternate keys to build indexes that identify records in alternate sort orders.”

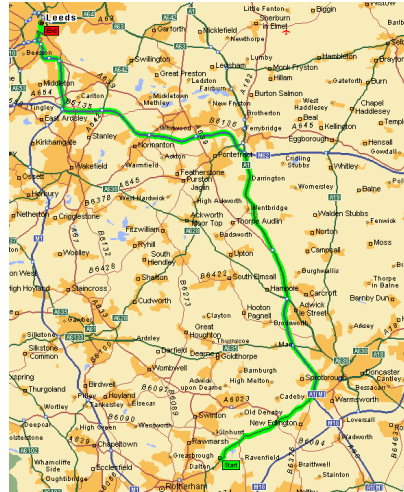
If you are used to using the terms *key* and *index* interchangeably, use this analogy to remind you of their difference:

Key is to destination

as

index is to route map

In practice, even if the terms are used loosely, the context makes the meaning clear.



Record Access Mode

Record access mode specifies how records are retrieved from and inserted into a file. Two record access modes are provided by RMS. They are sequential and random. Random can be subdivided, so that altogether there are four access modes:

- **sequential**
In sequential access mode, storage and retrieval begins at a designated starting point in the file and continues sequentially through the file. RMS begins accessing the records from the start of the file unless the starting point is explicitly specified, or is established in a previous operation.
- **random access by key value**
In random access mode it is the program and not the file organisation that determines the record processing order. For example, to randomly access a record by key value (in an indexed file) the program must provide the appropriate key and the key value.
- **random access by relative record number**
The program determines the record processing order. For example, to randomly access a record by relative record number in a sequential file with fixed length records the program must provide the relative record number.
- **random access by record file address (RFA)**
Each record on a disk has a unique file address known as the record file address. The RFA remains constant for as long as the record remains in the file. Whenever a program stores or retrieves a record, RMS returns the RFA to the program. The program can then either use this RFA as a random-access pointer to the record for subsequent access, or it can ignore the RFA.

File organisation and access mode are closely related but distinct from each other. The file organisation is established when the file is created; it defines the physical arrangement of records in the file. The organisation of a file cannot be changed unless you use a utility conversion routine (such as the Convert utility) to create the file again with a different organisation. However, the record access mode can be changed each time a record is accessed, within the constraint that the access mode is supported by the file organisation. The table below shows which access modes are permitted with each of the three file organisations.

Access Mode	File Organisation		
	sequential	relative	indexed
sequential	yes	yes	yes
random access by key value	no	no	yes
random access by relative record number	yes (fixed length record format on disk devices only)	yes	no
random access by record file address (RFA)	yes (disk devices only)	yes	yes

Note - Block I/O

Block I/O allows the bypass of the record processing capabilities of RMS. A file can be accessed as a contiguous set of blocks (say if you knew that you wanted the first 10,000 characters only). A program reads or writes one or more blocks by specifying a starting virtual block number in the file and the length of the transfer. RMS accesses the requested blocks regardless of the file organisation.

RMS files contain internal information which has meaning only to RMS. Digital recommends that you do not modify an existing file using block I/O if that file is to be accessed by record-level operations.

Note - ISAM

Indexed files are often processed by locating a record directly by key, and then using that key's index to sequentially read all the indexed records in ascending order of their key values. The term *indexed-sequential access method* (ISAM) is used by Digital to refer to this method of record access.

GLOSSARY

Random access An access mode in which the value of a data item identifies the record.

Sequential access An access mode in which records are obtained from a file in a consecutive sequence.

Record Format

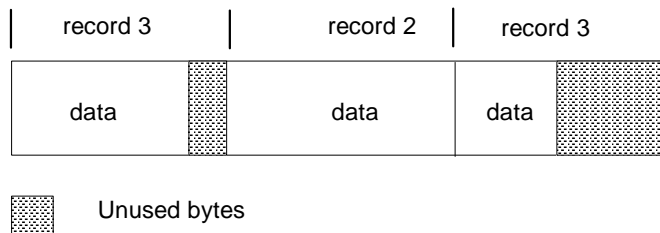
In general, RMS is more concerned with the format of a record (i.e. the physical structure on the storage medium) than it is with the content of a record. The exception to this is the key values that are part of the records in indexed files.

Four record formats are supported by RMS, namely

- fixed-length format
- variable-length format
- variable-length with fixed length control field (VFC) format
- stream format

These are described in more detail below.

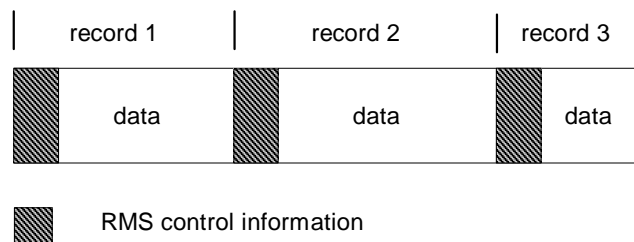
Fixed-length format



All the records in the file are the same length. The record length is set at file-creation time and becomes part of the information that RMS stores and maintains for the file. The specified length must be able to accommodate the longest record in the file. If any record fields are not used, a program must be able to detect them and provide appropriate error processing.

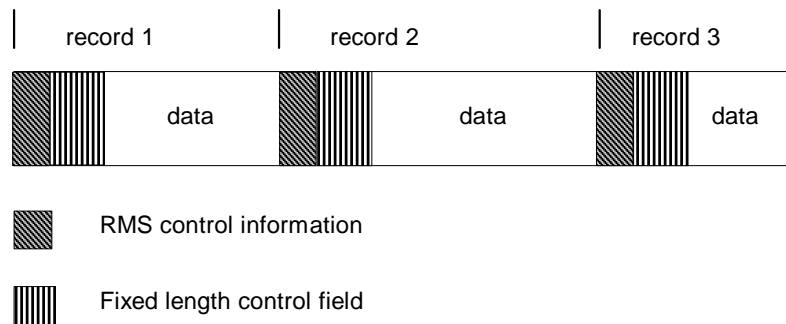
Each record must start on an even byte boundary.

Variable-length format



Each record is as long as the data within it requires, except that all records are padded to an even number of bytes. For example, if the data were 171 bytes, then 172 bytes would be written to the file - 171 for the data, and one byte padding. The number of bytes needed to store the data is encoded in a count field prefixed to the record. For disk files, this count field is two bytes long, and starts on an even byte boundary (which explains the need for the padding in the data record). RMS builds the count field from information in your program and treats it separately from the associated record data field.

VFC format



Variable-length with fixed length control field (VFC) format records are similar to variable-length records except that a fixed-length control field is placed before the variable-length data portion. This control field can contain additional data that has no direct relationship to the data itself. For example, it could contain line sequence numbers. The size of the control field is specified when the file is created.

Stream format

In a stream formatted file RMS treats the data as a continuous stream of bytes, without control information. There are four variants. Special characters or character sequences called terminators delimit the records in files in the first three variants of stream record format.

- **STREAM-CR** uses a carriage return as the terminator
- **STREAM-LF** uses a line feed as the terminator
- **STREAM** this variation ignores leading zeros, and uses one of these four combinations to delimit records
 - carriage return (CR)
 - carriage return / line feed (CR/LF)
 - form feed (FF)
 - vertical tab (VT)
- **UNDEFINED** this variation has no terminator; the length of each record is determined by the size of the buffer within the application program, or the end-of-file.

The table below shows which record formats are permitted with each of the three file organisations.

Record Format	File Organisation		
	sequential	relative	indexed
Fixed-length format	yes	yes	yes
Variable-length format	yes	yes	yes
Variable-length with fixed length control field (VFC) format	yes	yes	no
Stream format	yes	no	no

File Protection

File protection is the scheme used by OpenVMS to safeguard files against unauthorised access or modification. In this section we describe file protection in abstract terms.

You can protect a file in two ways:

- UIC-based protection codes
- ACLs (Access control lists)

This section focuses on the first method, as it is the most common. Both these protection methods also apply to directories and devices, but we won't consider these.

UIC-Based Protection

Every OpenVMS user has a *UIC* (user identification code) which is assigned by the system manager. There are two formats for UICs - numeric and alphanumeric. A numeric UIC consists of a group identifier and a member identifier which are separated by a comma and which are enclosed in square brackets. An alphanumeric UIC consists of a member name and an optional group name.

UICs cannot be arbitrarily assigned. There are two important rules:

- Member names must be unique for each user in the system (and typically, member name is the same as the user login name)
- No member can be in more than one UIC group

A UIC is specified in the user's record in the *user authorization file* which is a file containing an entry for every user that the system manager authorises to gain access to the system. Each entry identifies the user name, password, default account, UIC, quotas, limits and privileges assigned to the individuals who use the system.

Every file and directory is owned by a particular UIC, which is its *owner UIC*. Normally, this is the UIC of the person who created the file. When a user attempts to access a file, their UIC is compared with the owner UIC, and depending upon the relationship of the UICs, the user is placed in one or more of these categories.

- System (refers to operating system, system users and system manager)
- Owner (refers to the particular member of a group to which the file belongs.)
- Group (refers to a set of users who have identical access privileges)
- World (refers to all users, including system operators and users in the owner's group and any other group)

The relationship between the owner UIC of a file and the UIC of the user who wants to access the file determines whether that access is permitted or not. Depending on your classification, you may be allowed or denied the following types of access:

- Read Can examine, print, or copy a file
- Write Can modify or write to a file
- Execute Can execute a file that contains a DCL command procedure or an executable program image
- Delete Can delete a file

The allowed operations that can be performed by members of each of these groups is specified by the *protection code* (this is also sometimes known as the *protection mask*). This can be set when the file is created, and it can also be modified later.

For each of the four access categories the protection code specifies which of the four access operations is allowed. A protection code is specified and displayed in the following format.

(system: *rwed*, owner: *rwed*, group: *rwed*, world: *rwed*)

The access categories are listed from highest to lowest. In each category, the letters indicate the following types of access:

r = read
w = write
e = execute
d = delete

A letter is present if the corresponding access is allowed in the category, and absent if it is not. An example is shown below.

```
$ SHOW SECURITY TESTFILE.DAT
```

```
DKA300:[MIKEU]TESTFILE.DAT;1 object of class FILE
```

```
  Owner: [MIKEU]
```

```
  Protection: (System: RWED, Owner: RWED, Group: RE, World)
```

```
  Access Control List: <empty>
```

```
$
```

ACL-Based Protection

An ACL is a list of people or groups who are allowed to access a particular file. ACLs offer more scope than UICs in determining what action you want taken when someone tries to access your file. You can provide an ACL on any file to permit as much or as little access as you want.

You can specify the ACL for a file when you create it if you use RMS directly, but you cannot specify an ACL in an FDL specification. After a file is created, you can define the access control list for it with the ACL Editor.

According to DEC, many users do not need to bother with ACLs. However, ACLs allow greater control, and so typically are used by system managers and security administrators.

GLOSSARY

ACL (Access Control List) A list that defines the kinds of access to be granted or denied to users of an object.

Owner UIC The UIC of the owner of the file. Typically, the person who created the file.

Protection code A code in each file that indicates who may and may not access the file.

UIC (User Identification Code) A unique identifier for a user in the system.

Summary

In this module you have learnt

- ✧ RMS provides an interface to record and file management functions
- ✧ The different ways of accessing files (Record access mode)
 - Sequential. Retrieval begins at a designated starting point in the file
 - Random. Access to any record by key value
- ✧ There are three RMS file organisations
 - Indexed. This allows both random and sequential access to records. It is the most complex of the file organisations
 - Relative. Records are stored in fixed size cells
 - Sequential. This is the only organisation applicable to tape devices
- ✧ File organisation and record access mode are related, but different from each other
- ✧ Four record formats are supported by RMS
 - Fixed. All records are the same length
 - Variable. Each record is as long as the data within it requires
 - VFC. Similar to variable, but with fixed-length control field
 - Stream. Terminators delimit the records
- ✧ In an indexed file organisation keys are defined on records
 - Primary key - this must always exist
 - Alternate keys - these are additional supplementary keys
- ✧ A key can be
 - Simple - defined over one contiguous section (segment) of a record
 - Segmented - defined over two or more non-contiguous segments
- ✧ Indexes are used to access the records in an indexed file
 - Primary - must always exist. Uses the primary key to access records
 - Alternate indexes - use alternate keys to access records
- ✧ File attributes (physical characteristics) are defined when a file is created e.g. name, size, location, record format
- ✧ File protection is used to limit the operations that different users can perform on your files

RMS Review Questions

1. What is RMS and what does it do?
2. What are the three RMS file organisations?
3. What are the two basic record formats?
4. What is record access mode?
5. There are four different record access modes which fall into two basic categories. What are these two categories?
6. What's the difference between file organisation and record access mode?
7. Can all of the record access modes be used with all of the file organisations?
8. Which file structures allow random access based on the content of a record?
9. In RMS terms, what is a primary key?
10. In RMS terms, what is an alternate key?
11. What is a segmented key?

CURRENCY_RATE.DAT

There is an RMS data file, CURRENCY_RATE.DAT, which contains information on currency rates. It is an indexed file with fixed length records. The structure of the records has been defined as:

CURRENCY_CODE 4 BYTES	CURRENCY_RATE 8 BYTES	CURRENCY_DATE 11 BYTES
--------------------------	--------------------------	---------------------------

The data that is contained in the CURRENCY_CODE field is not unique; there will be many instances of the same code. CURRENCY_RATE contains numeric data. CURRENCY_DATE is stored as ASCII text.

Exercise

1. If you were going to use just simple keys, what do you think the primary key should be?
2. And what do you think would be a good alternate key?
3. What's the record size?
4. Would you consider using a segmented key? Why?

Model answers on page 3-28.

You will see the CURRENCY_RATE.DAT file again in later modules.

RMS Review Answers

1. What is RMS and what does it do?

RMS stands for Record Management Services. It is a data management facility provided by OpenVMS. It interprets the record structure within a file, and it provides routines which are used to manipulate files.

2. What are the three RMS file organisations?

Sequential, Relative, Indexed.

3. What are the two basic record formats?

Fixed length and variable length.

4. What is record access mode?

The method used in RMS for retrieving and storing records in a file.

5. There are four different record access modes which fall into two basic categories. What are these two categories?

Sequential, random (Digital also call this *direct*).

6. What's the difference between file organisation and record access mode?

File organisation is the structure of the file; record access mode describes how the data is accessed.

7. Can all of the record access modes be used with all of the file organisations?

No. The structure of a file restricts the record access mode.

8. Which file structures allow random access based on the content of a record?

Indexed. One or more keys is defined on the relevant fields of the record.

9. In RMS terms, what is a primary key?

It is the key which must always be defined in an indexed file.

10. In RMS terms, what is an alternate key?

It is an additional key which is defined in order to provide additional (to the primary) indexing.

11. What is a segmented key?

A key which is defined over two or more non-contiguous sections of a record.

Exercise Model Answers

There is no one definitive answer to each of these questions (apart from question 3!). To a great extent the right answer depends upon how the data will be accessed. A few pitfalls are highlighted here.

1. If you were going to use just simple keys, what do you think the primary key should be?

If you wanted to access the data based on currency, then use CURRENCY_CODE.

If you wanted to access the data based on the date, then use CURRENCY_DATE. However, this would only be appropriate if you wanted to access data based on a specific day. The reason is that the data is stored as ASCII text in this file. For example, imagine that there were three data records with these date values:

```
21-FEB-1997
20-JUN-1997
19-DEC-1997
```

If you wanted to return all data based on or after 20th June 1997, you would expect to see two records - one for 20th June and one for 19th December 1997. If you used a comparison based on the values of the data fields, your RMS program or ODBC application would return the wrong results:

<u>expected result</u>	<u>actual result</u>
20-JUN-1997	20-JUN-1997
19-DEC-1997	21-FEB-1997

This is because in this particular case the date is stored as ASCII text; 21-FEB-1997 is greater than 20-JUN-1997, and similarly, 19-DEC-1997 is less than 20-JUN-1997.

All the data for days 01 would be returned before all the data for days 02 and so on. Then within each day there would be an alphabetic ordering of the month, so that April would be the first month, and September the last one.

Note - reversed byte order

The problem of dates being returned in the wrong order (in terms of its real-world meaning) can be overcome by storing the data in reversed byte order, if the date is represented as a number.

Say that there is a date data type which has the format DD-MM-YYYY. For example, 17th March 1997 would be shown as 17-03-1999. If data values were stored in reversed byte order, for example, 1999-03-17, then when a range of data values are accessed using an index, they will be returned in the correct order. (In the case of CURRENCY_DATE described above, reversed byte order would not be sufficient in its own right, since there would still be the problem of the months being returned in alphabetic order).

2. And what do you think would be a good alternate key?

Again, the answer to this question depends upon how you intend to retrieve the data. We can imagine that in most situations, data may be retrieved based upon either the currency code, or the date (and of course, a combination of both - see question 4).

Within the constraints described in the answer to question 1, CURRENCY_DATE is a suitable alternate key (assuming you chose CURRENCY_CODE as the primary key).

3. What's the record size?

23 bytes

4. Would you consider using a segmented key? Why?

This all depends upon how the data will be accessed. It is likely that users will want data based on some currency code and date combination. Therefore, a segmented key using these two fields would be appropriate, if the data will be accessed based on a single date. Use a segmented key on CURRENCY_CODE and CURRENCY_DATE (in place of a simple primary and a simple alternate key). There will be no duplicates in the segmented key.

However, if the data will be accessed based on a date range, then using an index on the date field will not help.

If the file had this structure, you could define a simple primary key over the first 15 bytes.

CURRENCY_CODE 4 BYTES	CURRENCY_DATE 11 BYTES	CURRENCY_RATE 8 BYTES
--------------------------	---------------------------	--------------------------

Bibliography



This bibliography lists a few sources of information relevant to information in this module. Entries are listed alphabetically by title, except that initial articles (i.e. 'A' and 'The') are ignored for the purposes of ordering.

A Beginner's Guide to VAX/VMS Utilities and Applications, R. M. Sawe and T. T. Stokes, Digital Press, 1992. ISBN 1-55558-066-1.

The Digital Dictionary (second edition), R. E. Marotta (editor), DECbooks, 1984. ISBN 0-932376-82-7

A comprehensive and definitive reference to the terminology used with DEC systems. Also includes generic terms in computing.

Guide to OpenVMS File Applications, Digital Equipment Corporation, 1994.

VAX/VMS Internals and Data Structures: version 5.2, R. E. Goldenberg and L. J. Kenah, Digital Press, 1991. ISBN 1-55558-059-9.

Good glossary found in chapter 1 - System Overview.

Writing Real Programs in DCL, P. C. Anagnostopulos, Digital Press, 1989. ISBN 1-55558-023-8.

VMS File System Internals, K. McCoy, Digital Press, 1990. ISBN 1-55558-056-4.

The VMS User's Guide, J. F. Peters, P. Holmay, Digital Press, 1990. ISBN 1-55558-014-9.

4. The Easysoft System (RMS)

This module presents the global architecture of Easysoft ODBC for RMS.

In this module you will learn about

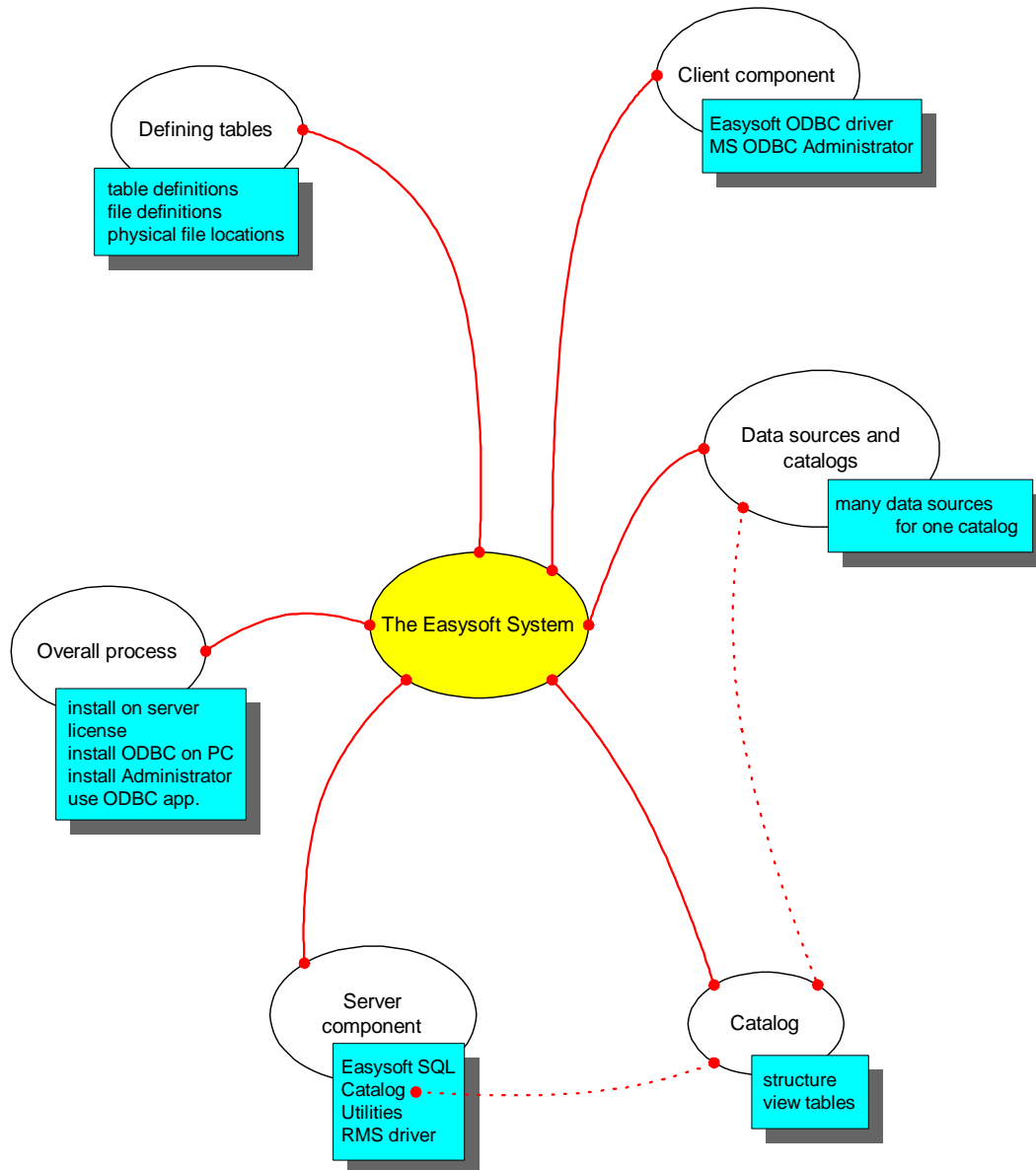
- ✦ the architecture of Easysoft ODBC for RMS
- ✦ the overall procedure for using Easysoft software
- ✦ the relationship between data sources, catalogs, databases and physical files
- ✦ the mapping between files and tables
- ✦ the tables in the Easysoft catalog
- ✦ the steps involved when data is server accessed by an ODBC application

At the end of the module there are some review questions.

Contents

Easysoft Architecture _____	4-3
Overall Process of using Easysoft _____	4-4
Data Sources and Catalogs _____	4-5
Defining Tables _____	4-6
Easysoft Catalog Structure _____	4-9
View the Catalog Tables _____	4-10
Accessing Server Data _____	4-15
Summary _____	4-16
Review Questions _____	4-17
Review Answers _____	4-18

Easysoft System Mind Map

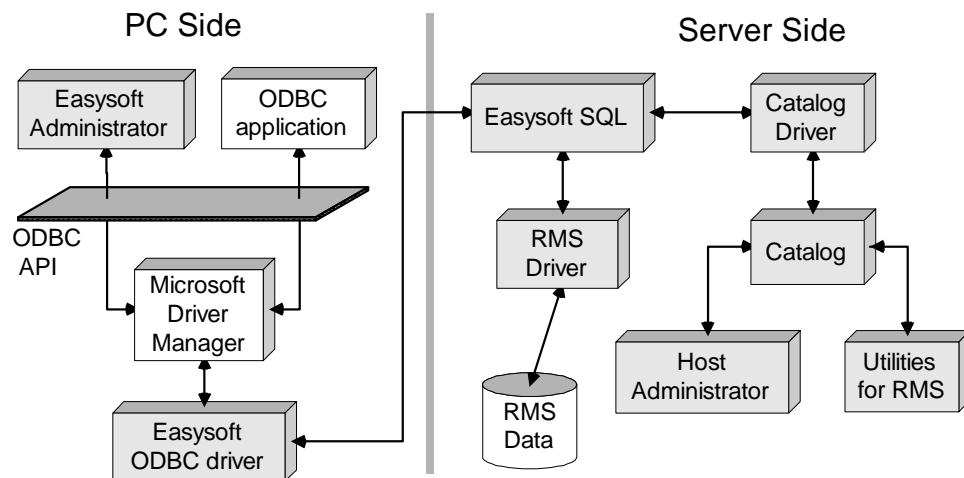


Easysoft Architecture

Easysoft ODBC for RMS is used to connect ODBC-compliant applications (such as Lotus 1-2-3 and Microsoft Excel) to RMS data files which reside on remote file Servers (on the OpenVMS platform). Since RMS data is not relational (i.e. not perceived to be in tabular form), one of the functions of the software, in addition to its ODBC functionality, is to make RMS data appear relational, so that ODBC-compliant applications can read from and write to RMS data.

Easysoft ODBC has a two-tier architecture in that the data access software resides on the Server and the driver passes function calls to this. The figure below shows the logical architecture of the components of the system and their relationship within the overall ODBC architecture.

Easysoft ODBC for RMS software architecture



Easysoft Client Component. This resides on the PC and contains the Microsoft ODBC Administrator and the Easysoft ODBC driver. The Microsoft ODBC Administrator is used to install the Easysoft ODBC driver on the PC and to configure data sources.

Easysoft PC Administrator. This contains the software which is used on the PC to define Server files for use through the Easysoft ODBC driver. This software presents Server files in a relational format; it maps files to tables and fields to columns. In addition to various utility functions, the Administrator is used to define users and security privileges.

Collectively, the software on the server is known as the **Easysoft Server Component**. This contains a number of sub-components:

Easysoft SQL - deals with processing of SQL statements.

Host Administrator - runs on the host machine and is used to manage catalogs at a high level - primarily their creation. It is also used to administer licensing functions.

Catalog - Easysoft SQL needs a set of files in order to manipulate the data which resides on the server. Collectively, these files are known as the Easysoft catalog. They hold, for example, information on the location and structure of data files.

Catalog Driver - the mechanism for obtaining information on the structure of the data.

RMS Driver - used to access RMS data on the Server (this should not be confused with the Easysoft ODBC driver which resides on the PC).

RMS Utilities -used as an aid to accessing data e.g. data dictionary converters.

Overall Process of using Easysoft

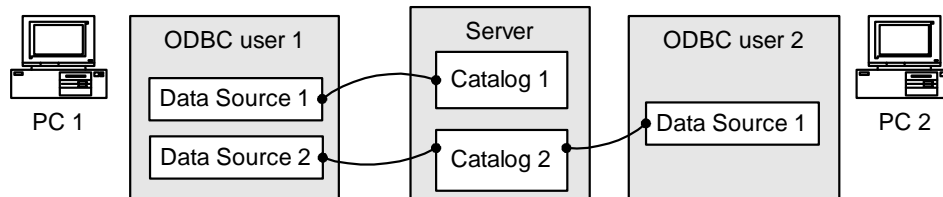
How exactly is the Easysoft system told of the existence and structure of the data files on the Server, and what do you do to enable an ODBC-compliant application to access this data? The recommended procedure is:

1. Install the Easysoft Server Component on the Server.
2. License the software.
3. Install the Easysoft Client Component on the PC and then define one or more data sources associated with a catalog.
4. Install the Easysoft Administrator on the PC and then define one or more databases for each of the data sources defined in step 3. For each database define the file to table mappings.
5. Use an ODBC-compliant application to access the data.

Data Sources and Catalogs

The figure below indicates the relationship between data sources and catalogs.

Relationships between data sources and catalogs

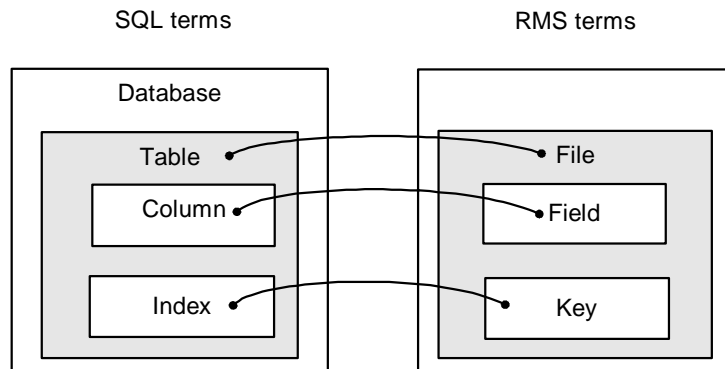


Data sources are defined at the PC level using the Microsoft ODBC Administrator. Each data source on a PC has a unique name. The data source defines which Server is to be accessed, the ODBC driver to use when accessing the Server data, the catalog associated with the data, network transport information and the user of the Server.

To ensure ease of upgrading it is advisable to keep data source names on different PCs identical if they are to be used to access the same catalog.

The correspondences between SQL and RMS are shown in the diagram below.

Correspondences between SQL terms and RMS terms



SQL tables map to files and SQL columns map to fields. A database is a uniquely named collection of SQL tables, and before the mapping between files and tables can be made, there must be at least one defined database. Associated with tables and databases are privileges which allow access rights to be defined. The function of the catalog is to hold all this mapping information.

Defining Tables

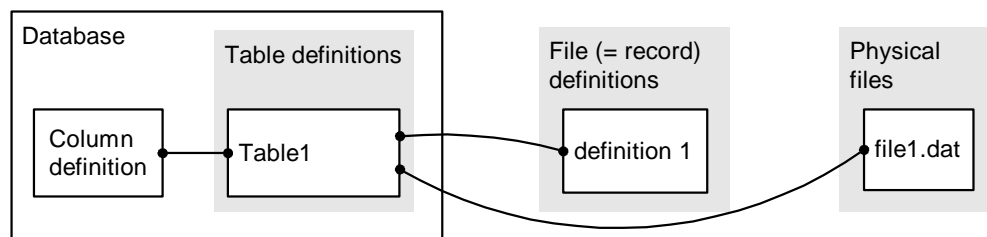
The basic entities to consider when defining SQL tables are the physical files residing on disk, file definitions and table definitions. These definitions are created using the Easysoft PC Administrator - you will use this in a later module.

Each database consists of a set of SQL tables. Each table is defined using a *table definition*. Each table definition links an SQL table to a physical data file (referenced by a file specification). The structure of the data file is described by means of a *file definition*, which is essentially a description of a record structure. File definitions and file specifications are not tied to any one database; a single file specification can be used in many different databases.

Each table consists of a set of columns. A *column definition* maps the available fields in the data file to the columns in the table. It is not necessary for all of the available fields to be used as columns in the table. Column definitions are not discussed further here, but you will use them in a later module.

The mapping between physical files and SQL table definitions is not complex, although there are numerous options. At first sight, there appears to be redundancy; one file definition can be used in many table definitions, one file definition can map (via a table definition) to many physical files, many table definitions can use the same file specification (i.e. physical file) and different file definitions can map onto one physical file. We now look at why there is not just simply a 1:1 mapping between file definitions and physical files.

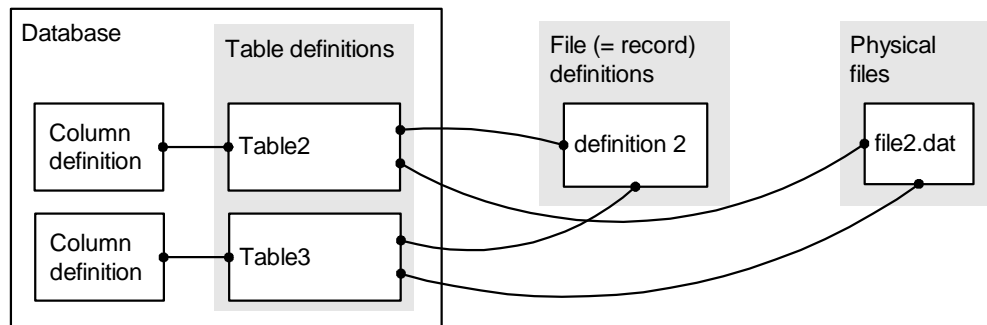
One file maps to one table



Look at the table definition above. One table definition references one file definition and one physical file. This is straightforward; a single file definition is used to define the structure of the physical file.

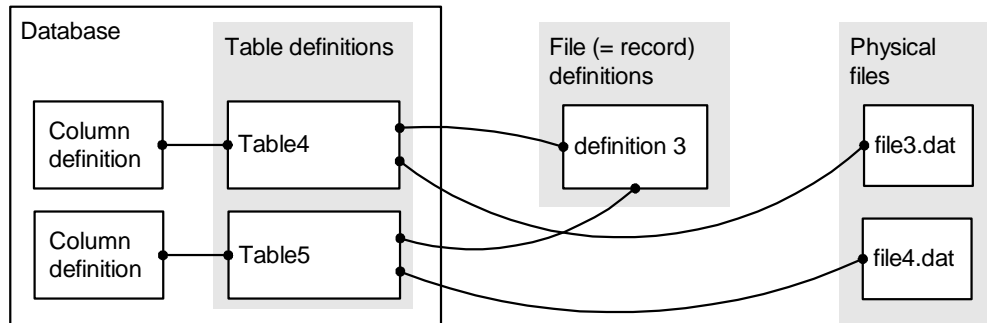
One file maps to two tables

More complex mappings are possible. Consider table2 and table3 below.



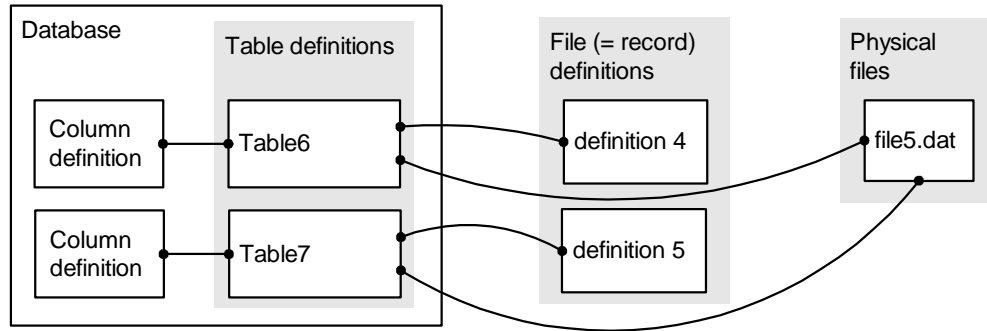
Here, the File Definition (i.e. definition of record structure) for both tables is the same and the file specification also is the same (file2.dat). The reason for this arrangement is that the column definitions for table2 and table3 can be different so that different users have different views of the data.

Different files have same record structure



Now consider the tables called table4 and table5 above. Both use the same file definition, but they access different physical files. This implies that the physical structure of file3.dat and file4.dat is identical in the parts that are defined. One situation where this might arise is where one file is used in earnest for company data and the other is used for development purposes. Another possibility is that there are separate files for, say, data which is grouped on a yearly basis. An example which demonstrates that an entire file structure need not be defined is where, say, the first n fields are identical, but fields $n+1$ upwards are undefined for ODBC purposes and are possibly different.

Different file definitions used on the same file



Finally, consider the situation pictured above. Here one physical file has two different file definitions. Each of these definitions is used by a different table definition. The end result of this is the same as the case where two different table definitions use the same file definition and file specification. This case is another example showing that it is not necessary to fully define the fields in a file - you only need to define those you are interested in accessing. This configuration could be used, for example, where there is a non-relational file which contains both header and detail records. The header records could be defined in SQL terms using one file definition, and the detail records would be defined using another definition.

These are the basic options -the possible combinations are not discussed as the basic principles remain the same. One final comment is that in spite of all these examples, typically, there is a 1:1 relationship (using the mapping in Table Definition) between a file definition and a physical file.

GLOSSARY

Column definition A list of the columns in a table.

File definition A description of the record structure of a data file on the server.

Table definition A table definition is used to define an SQL table. It links file definitions with physical files, and uses column definitions to describe the columns in the table.

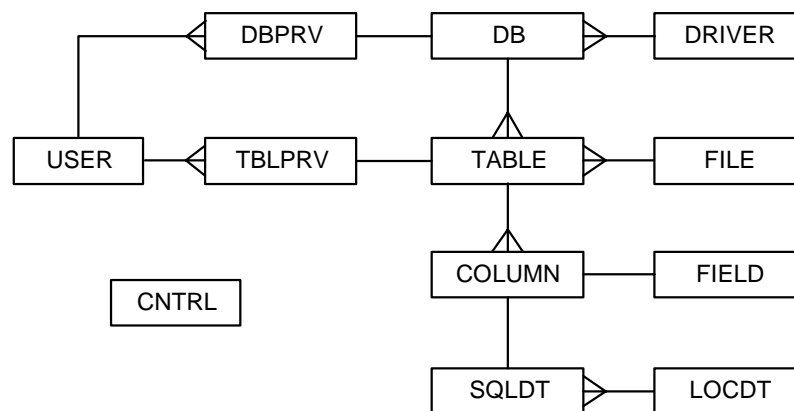
Easysoft Catalog Structure

Easysoft SQL needs a set of tables in order to manipulate the RMS data which resides on the Server. Collectively, these are known as the Easysoft Catalog. Data associated with a data source is defined in these Easysoft Catalog files, which hold, for example information on the location and structure of data files. A catalog can contain information on many different data sources.

You can have as many catalogs as you wish, but each catalog must reside in a separate directory. In most cases it is usual to have just one catalog to deal with all the data.

The diagram below shows the Easysoft catalog tables. (A crow's foot indicates a one-to-many relationship). The tables are contained in a database called ES, and so you cannot define a database of your own called ES.

Easysoft Catalog



The tables contain the following information. The definitions contained in these tables can be seen in the CSV (Comma Separated Values) text file which is exported from a catalog

ES_COLUMN	column definitions
ES_CNTRL	version information - not shown in CSV
ES_DB	database definitions
ES_DBPRV	database privileges
ES_DRIVER	driver information - not shown in CSV
ES_FIELD	field definitions
ES_FILE	file definitions
ES_LOCDT	local data types - not shown in CSV
ES_SQLDT	SQL data types - not shown in CSV
ES_TABLE	table definitions
ES_TBLPRV	table privileges
ES_USER	user definitions

The tables DRIVER, CNTRL, SQLDT and LOCDT contain information which is internal to the Easysoft system; the contents of these tables are not available in the CSV. However, you can still view the content of these tables - you will see how in the next section.

GLOSSARY

Catalog A collection of tables which contain definitions that describe features of a database, such as the columns in a table.

CSV (Comma Separated Values) Data values which are separated by commas.

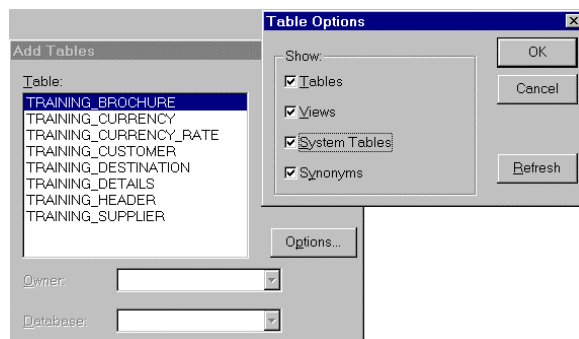
View the Catalog Tables

When you use an ODBC-compliant application to view the data on the server, you do not normally see the catalog tables. However, it is usually possible to view them (this depends on the application). Here we show how to do this using Microsoft Access and Microsoft Excel. The two methods are quite different. In Excel, you set an option whilst accessing the data. With Access, you set the option before you connect to the data. (The module entitled "Using ODBC Applications" contains step-by-step details of how to connect to a data source and retrieve data).

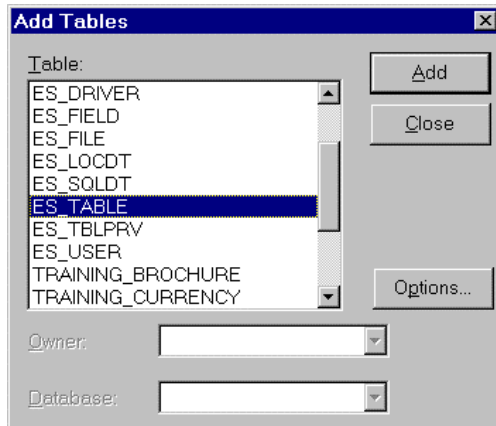


Microsoft Excel/Query

1. Start Excel. Select **Data, Get External Data**. The Select Data Source dialog box appears.
2. Select one of the data sources for the catalog you wish to view. Click the **Use** button. The Add Tables dialog box appears.
3. Click the **Options...** button. The Table Options dialog box appears. Click the **System Tables** option so that an arrow appears next to it.



- Click the **OK** button. The catalog tables now appear in the Add Tables dialog box. Select the ones you want and close the dialog box. (In this example, we used ES_TABLE. Notice that the database name which prefixes the table name is different for the catalog tables and the data tables. You will see how to define database names in the module entitled "Easysoft Administration on the PC").



- Click the **Add** button and then close the dialog box.
- Select all the columns and drag them into the Query pane. Then return the data to Excel.

	A	B	C	D	E	F	G	H
1	DATABASE NAME	TABLE NAME	TABLE TYPE	FILE NAME	FILE SPECIFICATION	NUMBER	COLUMN CRITERIA	REMARKS
2	ES	CNTRL	SYSTEM TABLE	CNTRL	ESCNTRL.DAT	4		System parameters
3	ES	COLUMN	SYSTEM TABLE	COLUMN	ESCOLUMN.DAT	22		Column details
4	ES	DB	SYSTEM TABLE	DB	ESDB.DAT	8		Databases available
5	ES	DBPRV	SYSTEM TABLE	DBPRV	ESDBPRV.DAT	12		Database privileges
6	ES	DRIVER	SYSTEM TABLE	DRIVER	ESDRIVER.DAT	5		Data source drivers
7	ES	FIELD	SYSTEM TABLE	FIELD	ESFIELD.DAT	14		Physical field desc.
8	ES	FILE	SYSTEM TABLE	FILE	ESFILE.DAT	8		Physical file desc.
9	ES	LOC DT	SYSTEM TABLE	LOC DT	ESLOC DT.DAT	6		Local data types
10	ES	SQ LDT	SYSTEM TABLE	SQ LDT	ESSQ LDT.DAT	18		SQL data types
11	ES	TABLE	SYSTEM TABLE	TABLE	ESTABLE.DAT	10		Table details
12	ES	TBLPRV	SYSTEM TABLE	TBLPRV	ESTBLPRV.DAT	10		Table privileges
13	ES	USER	SYSTEM TABLE	USER	ESUSER.DAT	5		User details
14	TRAINING	BROCHURE	TABLE	BROCHURE	BROCHURE.DAT	3		
15	TRAINING	CURRENCY	TABLE	CURRENCY	CURRENCY_MASTEI	4		
16	TRAINING	CURRENCY_	TABLE	CURRENCY_	CURRENCY_RATE.D	3		
17	TRAINING	CUSTOMER	TABLE	CUSTOMER	CUSTOMER.DAT	13		
18	TRAINING	DESTINATION	TABLE	DESTINATION	DESTINATION.DAT	2		
19	TRAINING	DETAILS	TABLE	DETAILS	INVOICE.DAT	6		TRAINING_DETAILS.LINE_NUMBER>0
20	TRAINING	HEADER	TABLE	HEADER	INVOICE.DAT	7		TRAINING_HEADER.LINE_NUMBER=0
21	TRAINING	SUPPLIER	TABLE	SUPPLIER	SUPPLIER.DAT	16		
22								

Data tables

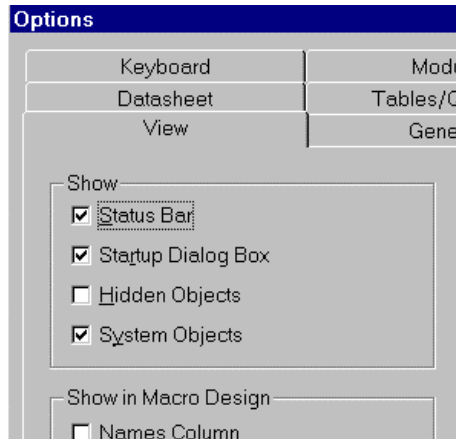
Catalog tables

- The Easysoft TABLE table shows all the tables known to the catalog, including the catalog tables themselves. Refer to the previous diagram to see how the catalog tables relate to each other. Look at a few of the different catalog tables in order to see the kind of information they contain.

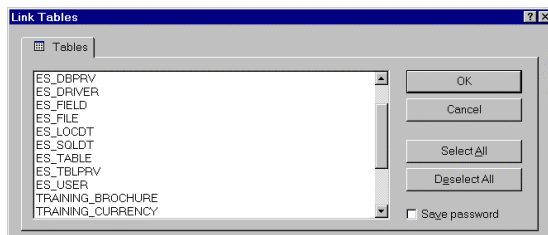


Microsoft Access

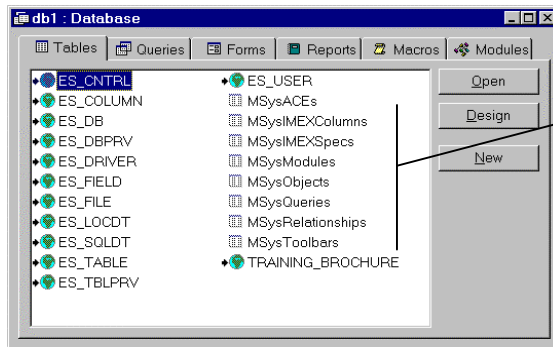
1. Start Access and open a database.
2. From the main menu bar select **Tools, Options...** The Options dialog box appears.
3. Select the **View** tab.



4. Select the **System Objects** check box.
5. Click the **OK** button.
6. Select **File, Get External Data, Link Tables**. The Link dialog box appears.
7. From File of type list box select **ODBC databases**. The Select Data Source dialog box appears. Select one of the data sources for the catalog you wish to view and click the **OK** button.
8. The Link Tables dialog box appear. Notice that it contains the catalog tables in addition to the data tables.
9. Select the tables you want and close the dialog box. (The database name which prefixes the table name is different for the catalog tables and the data tables. You will see how to define database names in the module entitled "Easysoft Administration on the PC").



10. Click the **OK** button. The catalog tables now appear in the Add Tables dialog box.



These tables are Microsoft Access system tables.

11. Double-click on a table to see its contents (here we use the ES_TABLE table)

DATABASE_NA	TABLE_NAME	TABLE_TYPE	FILE_NAME	FILE_SPECIFIC	NUMBER_COL	CRITERIA	REMARKS
ES	CNTRL	SYSTEM TABLE	ESCNTRL.DAT		4		System parameters
ES	COLUMN	SYSTEM TABLE	ESCOLUMN.DAT		22		Column details
ES	DB	SYSTEM TABLE	ESDB.DAT		8		Databases available
ES	DBPRV	SYSTEM TABLE	ESDBPRV.DAT		12		Database privileges
ES	DRIVER	SYSTEM TABLE	ESDRIVER.DAT		5		Data source drivers
ES	FIELD	SYSTEM TABLE	ESFIELD.DAT		14		Physical field desc.
ES	FILE	SYSTEM TABLE	ESFILE.DAT		8		Physical file desc.
ES	LOCDT	SYSTEM TABLE	ESLOCDT.DAT		6		Local data types
ES	SQLDT	SYSTEM TABLE	ESSQLDT.DAT		18		SQL data types
ES	TABLE	SYSTEM TABLE	ESTABLE.DAT		10		Table details
ES	TBLPRV	SYSTEM TABLE	ESTBLPRV.DAT		10		Table privileges
ES	USER	SYSTEM TABLE	ESUSER.DAT		5		User details
TRAINING	BROCHURE	TABLE	BROCHURE	BROCHURE.DAT	3		
TRAINING	CURRENCY	TABLE	CURRENCY	CURRENCY.M	4		
TRAINING	CURRENCY_R	TABLE	CURRENCY_R	CURRENCY_R	3		
TRAINING	CUSTOMER	TABLE	CUSTOMER	CUSTOMER.DAT	13		
TRAINING	DESTINATION	TABLE	DESTINATION	DESTINATION.DAT	2		
TRAINING	DETAILS	TABLE	DETAILS	INVOICE.DAT	6	TRAINING_DETAIL	
TRAINING	HEADER	TABLE	HEADER	INVOICE.DAT	7	TRAINING_HEADER	
TRAINING	SUPPLIER	TABLE	SUPPLIER	SUPPLIER.DAT	16		

12. The Easysoft TABLE table shows all the tables known to the catalog, including the catalog tables themselves. Look at a few of the different catalog tables in order to see the kind of details they contain.

Note: catalog content and CSV

There is not always a direct 1:1 correspondence between what you see in a table, and the CSV structure. The ES_TABLE table contains two columns that do not appear in the table definitions of an exported CSV file. NUMBER_COLUMN shows the number of columns contained in each of the tables (this can be calculated from the information contained in the COLUMN table). The other column is REMARKS.

Note: active files

You can easily determine which server files are currently being used (i.e. are active) by a query.

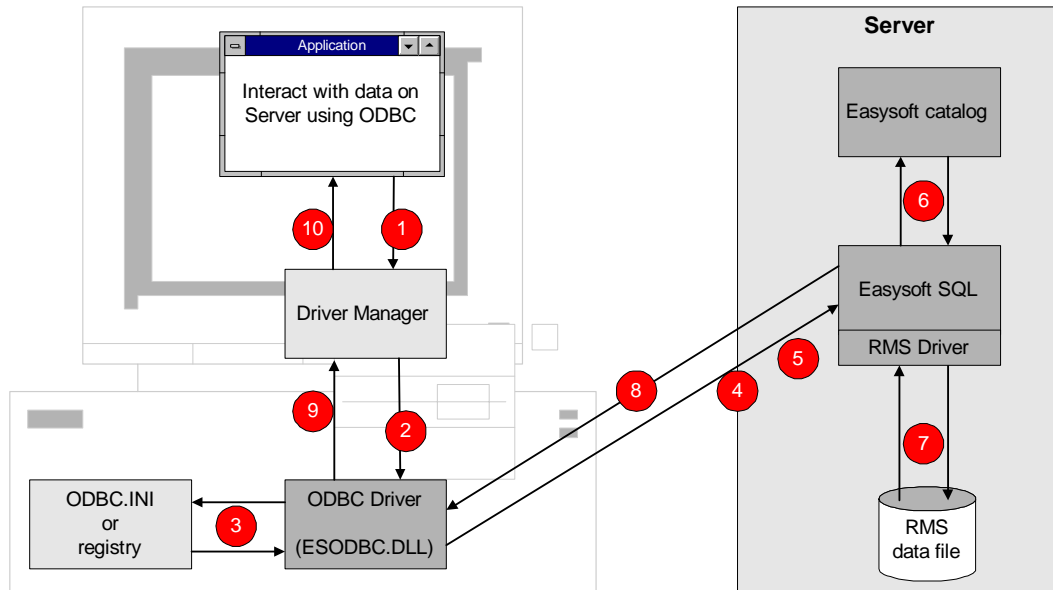
1. Log on to the server
2. View the files that are being used:
\$ **SHOW DEVICES /FILES**

You will see a list of active files

Accessing Server Data

The diagram below shows the main steps which occur when an application sends an ODBC call. The first stages involve establishing the ODBC connection (steps 1 to 4), and then the SQL statement generated by the application is sent to the server.

Main steps in accessing server data



The steps numbered on the diagram above are explained below:

1. An ODBC-compliant application sends an ODBC call to the (Microsoft) Driver Manager. The call includes information about the server login name and password and the catalog login name and password as well as the data source name.
2. The call is passed to the Easysoft ODBC driver. (The Driver Manager determines which driver to pass the call to by querying the ODBC.INI file (for Windows 3.x) or the registry (for Windows NT and Windows 95. The information about the driver was stored there when the data source was created or modified using the Easysoft ODBC Setup dialog box).
3. The Easysoft ODBC driver in turn queries the ODBC.INI file or the registry as appropriate in order to gather information about the server name and catalog location. The required information is returned to the Easysoft ODBC driver.
4. The server login information (i.e. OpenVMS username and password) is used to gain access to the server.

5. Now that a connection has been made, the SQL statement is sent from the application and routed through the Driver Manager to the Easysoft ODBC driver which in turn sends the SQL to the Easysoft SQL component.
6. Easysoft SQL uses the catalog to determine the location of the files and fields which correspond to the tables and columns specified in the SQL statement.
7. The Easysoft SQL engine in conjunction with the RMS Driver then uses RMS file operations to access the records in the RMS file which are needed to resolve the SQL query. The records are returned to Easysoft SQL which then converts these into the tabular SQL format.
8. Easysoft SQL then sends the result set back to the Easysoft ODBC driver.
9. The Easysoft ODBC driver passes the result set, along with any error messages, to the Driver Manager.
10. The Driver Manager forwards the result set to the application and also acts upon any error messages.

Summary

In this module you have learnt

- ✦ the Easysoft Client component contains the ODBC driver and resides on the PC
- ✦ a single PC can have many different data sources defined on it
- ✦ the Easysoft Server component consists of
 - Easysoft SQL, which processes SQL statements
 - the catalog, which contains a description of the RMS data
 - the Host Administrator, used mainly for licensing and catalog operations
 - the RMS driver, which accesses RMS data
 - the RMS utilities, such as data dictionary converters
- ✦ there are many options available for mapping data files to relational tables
- ✦ the structure of the Easysoft catalog
- ✦ how to view the Easysoft catalog using Access and Excel
- ✦ the steps involved in
 - sending SQL generated by an application to the server
 - returning the data to the application

Review Questions

1. Easysoft ODBC for RMS is
 - a) single tier
 - b) two-tier
 - c) three-tier

2. The data access software resides on
 - a) the PC
 - b) the server

3. Name the functions of the Easysoft Host Administrator

4. What is the function of the Easysoft catalog?

5. Where is the catalog stored?

6. Data sources are defined on
 - a) the PC
 - b) the server
 - c) either PC or server
 - d) both PC and server

7. Name the SQL equivalents of these RMS terms:
 - a) File
 - b) Field
 - c) Key

8. What is the overall process of installing and using Easysoft ODBC for RMS?

9. What must you do in an application to see the Easysoft catalog tables?

Review Answers

1. Easysoft ODBC for RMS is

two-tier

2. The data access software resides on

the server

3. Name the functions of the Easysoft Host Administrator

Create catalog, manage catalog, deal with licensing

4. What is the function of the Easysoft catalog?

Store information about RMS files

5. Where is the catalog stored?

On the server

6. Data sources are defined on

the PC

7. Name the SQL equivalents of these RMS terms:

RMS:	File	Field	Key
SQL:	Table	Column	Index

8. What is the overall process of installing and using Easysoft ODBC for RMS?

Install on the server

License the software

Install the client component on the PC and define a data source

Install the Easysoft PC Administrator define a database and tables

Access the data using ODBC-compliant application

9. What must you do in an application to see the Easysoft catalog tables?

Typically, you must select an option allowing you to see system tables (the name may vary). How you do this is application dependent.

5. Microsoft ODBC Administrator

Before you can use an ODBC-compliant application to connect to data on the Server, you must set up a data source. Later, when you use the application, you will connect to the server data using this data source. To set up a data source, use the Microsoft ODBC Administrator.

In this module you will learn

- ✦ how to create a data source using the Easysoft ODBC Setup dialog box

Contents

Purpose _____	5-2
Administrator _____	5-3
Set up a Data Source _____	5-4
Settings _____	5-6
Additional Notes _____	5-7
Summary _____	5-9
Supplement: troubleshooting _____	5-10

Purpose

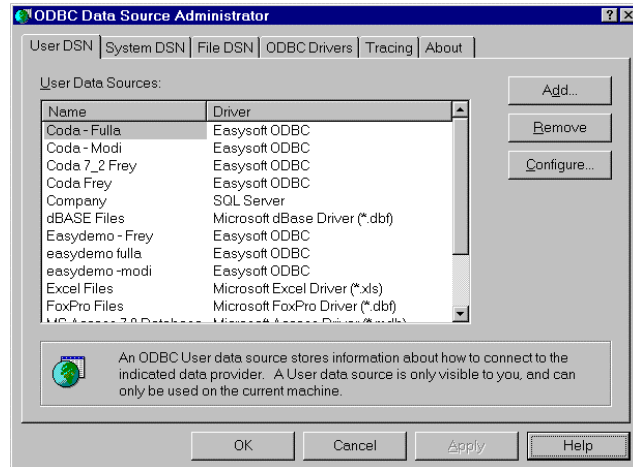
The Microsoft ODBC Administrator is used for the creation, deletion and modification of data sources in conjunction with driver-specific dialog boxes which are provided by the creators of the drivers. The creation of a data source is described in this module. The Administrator is also used for diagnostic purposes, namely the tracing of ODBC function calls (discussed in the appendix entitled, “Troubleshooting”).

This module gives step-by-step instructions for setting up (System) data sources, first using Microsoft ODBC Administrator.

Administrator

Version 3.0 is used with 32 bit drivers only. The layout of previous versions is significantly different, although the functionality is similar to that described here, namely to add, delete and configure data sources, and to set trace options.

Microsoft ODBC Data Sources Administrator



The following tabs are available on the Microsoft ODBC Data Source Administrator, version 3.0:

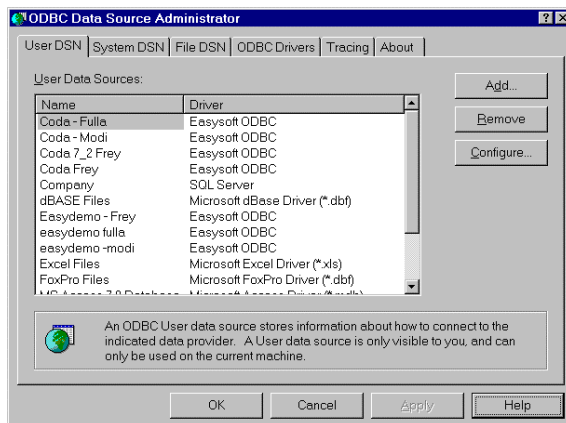
Tab	Function
User DSN	Add, remove and configure User data sources. A User data source is available to a specific user on a machine (compare System DSN).
System DSN	Add, remove and configure System data sources. A System data source is available to all users on a machine (compare User DSN).
File DSN	These are file-based data sources that can be shared between all users that have the same drivers installed. This option is not applicable to Easysoft ODBC and was not available in previous versions of the Microsoft ODBC Administrator.
ODBC Drivers	Displays information about the installed drivers (see "Drivers Tab").
Tracing	Set trace options for the ODBC Driver Manager. Described in Appendix G, "Troubleshooting".

About Information about the core components of the Microsoft ODBC. This About tab has no relationship to the About dialog box in version 2.5 of the Administrator, which described information about the installed drivers. For information about drivers, look under the ODBC Drivers tab.

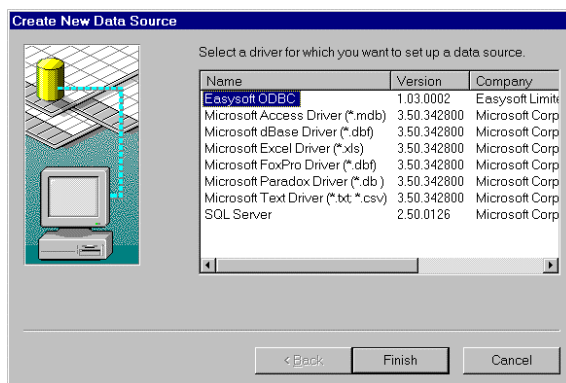
Set up a Data Source

The operation of the User data source dialog box and the System data source dialog box is identical. A System data source is created in this example.

1. Start the Microsoft ODBC Administrator version 3.0 by clicking on the ODBC icon (found in the Control Panel).
2. Select the **System DSN** tab.



3. Click the **Add...** button. The Create New Data Source dialog box is displayed.



4. Highlight **Easysoft ODBC** and click **Finish**. The Easysoft ODBC Setup dialog box appears. Complete this using the information below.

Enter this information

Reason / explanation

Name

TRAINING DATA

The name of a data source. This name appears in the ODBC connection dialog box when you connect to an ODBC source.

Naming restrictions:

- minimum number of characters = 1
- maximum number of characters = 32
- first letter must be alphabetic
- not valid: [] { } () ? * = ! @ , ;

Description

Type any description.

A more descriptive name for the data source (optional).

Catalog

This will be given on the day of the course.

The directory where the Easysoft Catalog resides on the Server.

Transport

Select **REXEC**.

The network transport to be used. You are presented with the options available to your particular PC.

Server

MODI

The name of the Server on which the source data resides.

Remote Service /Command/ Object

This will be given on the day of the course.

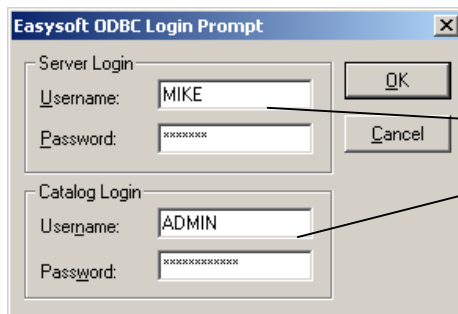
Select the name of the service to connect to on the Server. Remote Service and Remote Object map to a command to run the software. Remote Command is the command to run in order to access the software.

Usernames, Passwords

Leave blank.

If no user and password information is stored with a data source, each user has to enter this information in the Easysoft ODBC Login Prompt (see next figure) each time they connect.

5. For the purposes of this exercise, the **Settings**, **Logging** and **Capture** buttons are not needed. **Settings** is discussed in “Settings” on page 5-6, **Logging** is discussed in the appendix entitled “Troubleshooting”. (**Capture** is currently disabled).
6. Press the **Test** button. This validates the inserted information. The Easysoft ODBC Login Prompt appears, and the entry fields are initially blank. Information on how to switch off the login prompt is in the “Settings” section.



Server and catalog usernames and passwords can be different.

7. Enter the username and password information (you will be given these on the day of the course) and click the **OK** button. If the test is successful, a dialog box appears stating this; click **OK** to continue. If the test is unsuccessful a message is generated. After the test is successful, click on **OK** to save the new data source and return to the Data Sources dialog box. Leave the Data Sources dialog box by selecting **Close**.

Server Login Username The user name used to connect to the Server. Case is ignored.

Server Login Password The password associated with the Server login username. Case is ignored.

Catalog Login Username The user name that provides access to the catalog. Case is ignored.

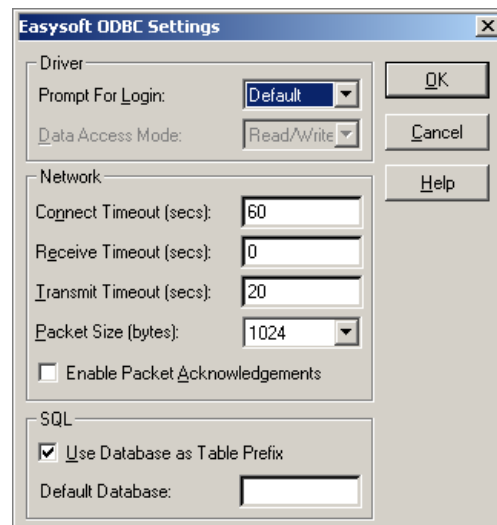
Catalog Login Password The password associated with the Catalog login user name. Case is ignored.

Now that a data source has been created, you can connect to data on the server using any ODBC-compliant application.

Settings

The Easysoft ODBC Settings dialog box is accessed by selecting **Settings...** on the Easysoft ODBC Setup dialog box. It is used to set driver, network and SQL options.

Easysoft ODBC Settings dialog box



Information required for Easysoft ODBC Settings

Prompt For Login	Controls the display of the Easysoft ODBC Login Prompt when a connection is made to a data source: Default: display is defined by the application. Never: dialog box never displayed. To ensure successful operation, ensure that login information is supplied in the Easysoft ODBC Setup dialog box. Always: dialog box always displayed.
Data Access Mode (not yet supported)	Allows data to be either read/write or read only.
Connect Timeout	The attempt to connect is cancelled if it is not successful within the specified time.
Receive Timeout	The connection is cancelled if a reply is not received within the specified time. Note: the value zero means wait forever.
Transmit Timeout	The connection is cancelled if the network layer cannot send the packet in the specified time.
Note: the timeouts for each option are identical for both Client and Server.	
Packet Size	Size of the data packets used to transmit data over the network. Valid options are 128, 256, 512, 1024 (default), 2048, 4096, 8192, 16384, 32768.
Enable Packet Acknowledgements	This option checks to see if data has been received. The default is OFF. If network problems are experienced it can be turned ON. Using this options slows down the transmission of the data.
Use Database as Table Prefix	In the case where a catalog contains information on two or more databases, and where those databases contain tables with the same name, this option allows us to differentiate the tables. e.g. If there are two SALES tables, one in a database called MINE and one in a database called THEIRS, then we would be able to connect to both of these using the data source. We would see the tables as MINE_SALES and THEIRS_SALES.
Default Database	If the Use Database as Table Prefix option is not selected, then a default database must be specified, even if only a single database exists. If there is more than one database in the catalog, only the one specified with this option will be accessible.

Additional Notes

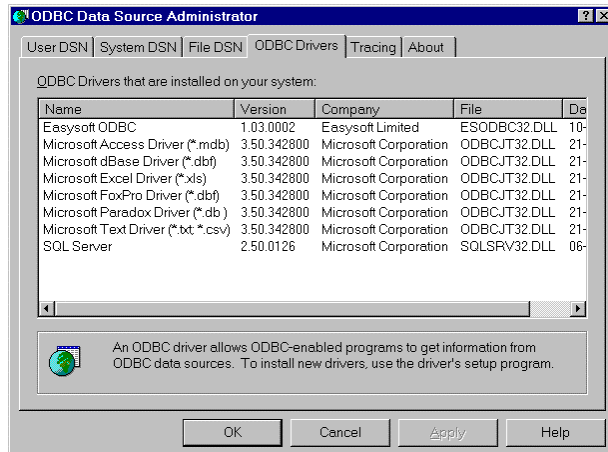
To remove a data source, highlight it and select **Remove**. A dialog box appears asking for confirmation. Choose **Yes** to delete the data source, **No** to cancel the operation.

To modify a data source, highlight it and select **Configure...**

Drivers Tab

The ODBC Drivers tab gives information about drivers.

ODBC Drivers tab



Each line in the list box shows the driver name, followed by the version, company, name of the driver file and the date it was produced.

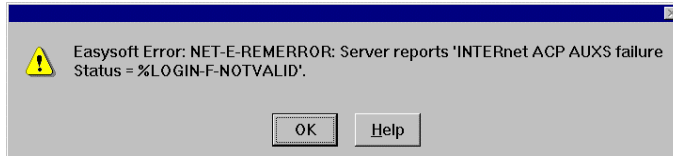
Summary

In this module you learnt

- ✦ the Easysoft ODBC Setup dialog box is used to add and modify Easysoft data sources

Supplement: troubleshooting

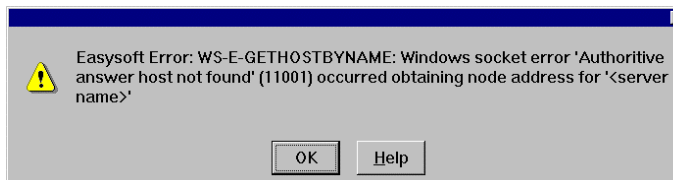
This section lists some of the common error message that can arise when connecting to server data. In cases where the answer is not obvious, a solution is described.



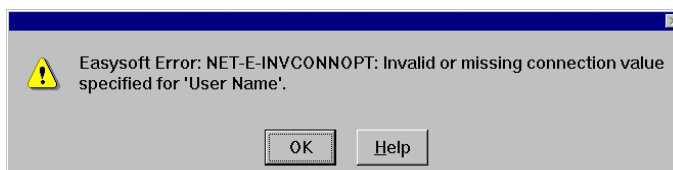
Cause: Incorrect server login name and/or password specified in the Easysoft ODBC Setup dialog box or the Easysoft ODBC Login Prompt.



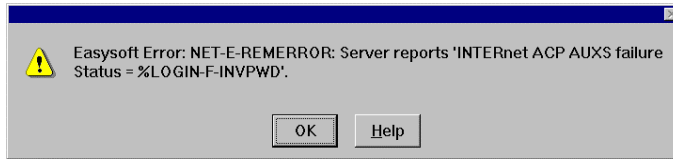
Cause: Incorrect catalog login name and/or password specified in the Easysoft ODBC Setup dialog box or the Easysoft ODBC Login Prompt.



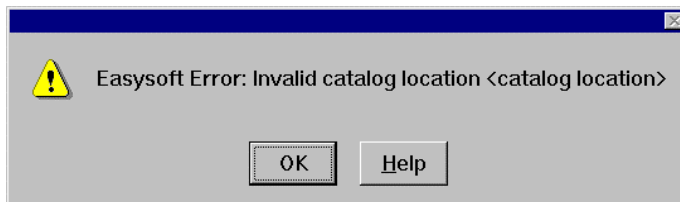
Cause: Incorrect server name specified in the Easysoft ODBC Setup dialog box or the Easysoft ODBC Login Prompt.



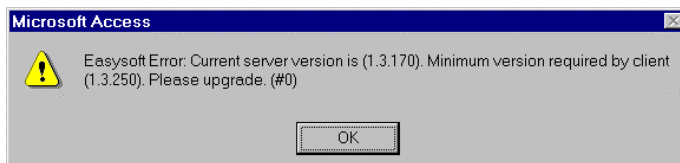
Cause: A blank user name was specified in the Easysoft ODBC Setup dialog box or the Easysoft ODBC Login Prompt.



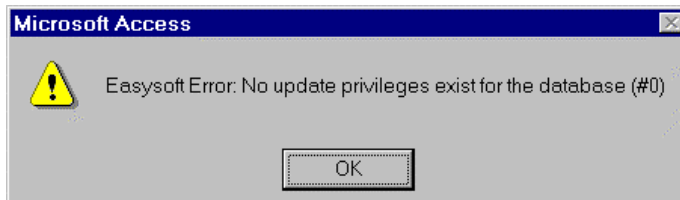
Cause: A blank password was specified in the Easysoft ODBC Setup dialog box or the Easysoft ODBC Login Prompt.



Cause: non-existent catalog location specified in the Easysoft ODBC Setup dialog box.



Cause: different versions of Easysoft on the server and the PC.



Cause: An attempt was made to update a record. However, the user does not have update privileges. Similar messages occur for other operations (select, insert and delete) if the user does not have the required privileges.

6. Using ODBC Applications

This module shows you how to read server data using popular ODBC-compliant applications. You will also see how to access the data directly from the server.

In this module you will learn how to

- ✦ download data using six popular ODBC-compliant applications
- ✦ add criteria to a query in each of the applications
- ✦ view the SQL statement that was sent in a query
- ✦ view available indexes using Microsoft Access

Contents

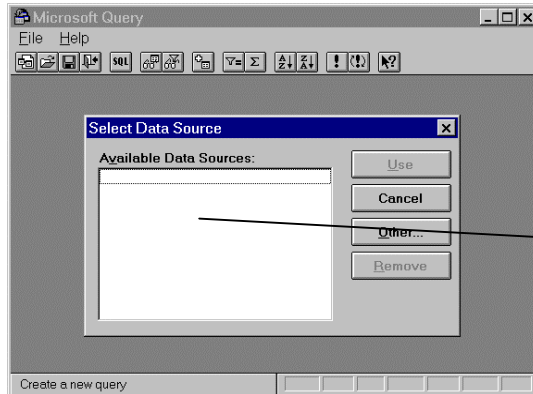
Microsoft Excel	6-2
Download Entire Table	6-2
Add Criteria	6-5
View SQL	6-7
Return Data to Excel	6-8
Microsoft Access	6-10
Download Entire Table	6-10
Add Criteria	6-12
View SQL	6-14
View Indexes	6-14
Lotus 1-2-3	6-15
Configure Lotus 1-2-3 to work with Easysoft ODBC	6-15
Download Entire Table	6-16
Add Criteria	6-18
View SQL	6-19
Microsoft Word Mail Merge	6-20
Impromptu	6-25
Preparation: Create an Impromptu Catalog	6-25
Download Entire Table	6-28
Add Criteria	6-29
View SQL	6-30
Crystal Reports	6-31
Download Entire Table	6-31
Add Criteria	6-34
View SQL	6-35

Microsoft Excel

These examples are based on Microsoft Excel, version 7.0, for Windows 95 and Microsoft Query, version 2.0. In these exercises you will download the entire contents of the CURRENCY_RATE table. Then you will set a criterion so that you see the rates for just one currency. Finally, you will see how to view the SQL that is sent to the server.

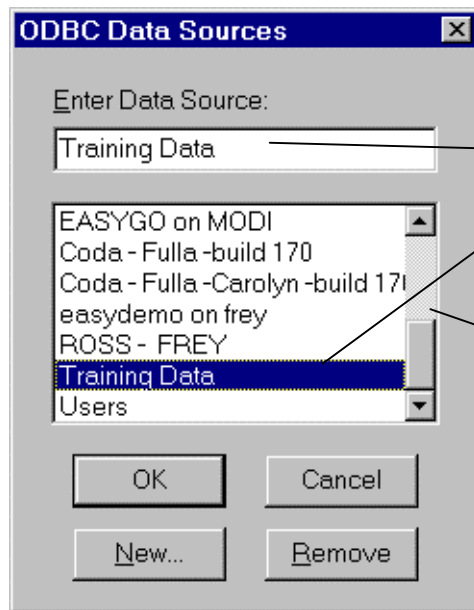
Download Entire Table

1. Start Excel and select an empty worksheet.
2. Select **Get External Data...** from the **Data** menu. Microsoft Query appears with the **Select Data Source** dialog box.



Previously used data sources appear here.

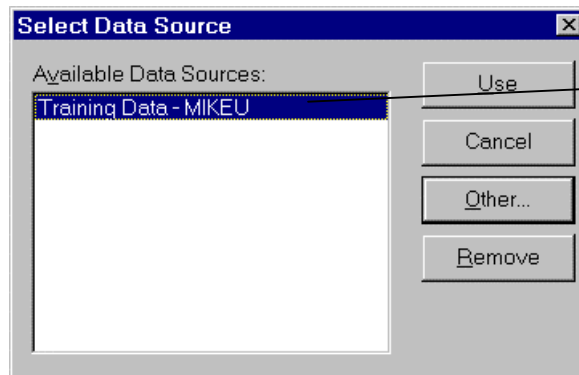
3. Since the data source has not been used before with Microsoft Query, it does not appear in the Available Data Sources list box, and so you have to make it available for use. Click **Other...**. The ODBC Data Sources dialog box appears. (In the general case, if the data source were to appear in the list, you would proceed directly to step 5).



Highlighting a data source causes it to appear in the top box.

Data sources are not in alphabetical order. Use the scroll bar to see all available options.

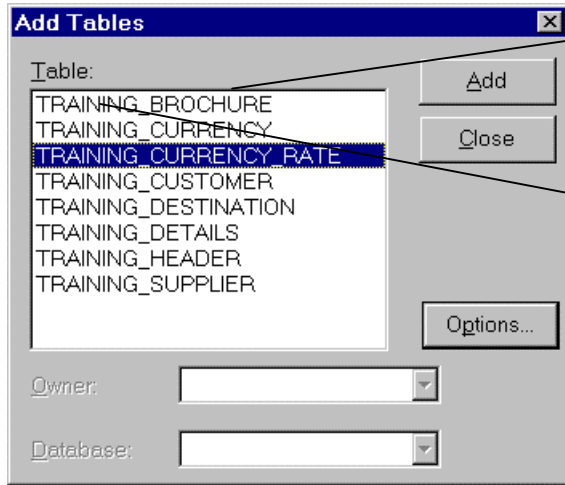
4. Select TRAINING DATA from the Enter Data Source list box and click **OK** when you have done this. Control passes back to the Select Data Source dialog box.
5. Highlight the TRAINING DATA data source.



The name of the data source may be amended automatically - here it is suffixed with the user name.

6. Click the **Use** button, and the Add Tables dialog box appears - this may take a few seconds. (In the background you are able to see changes to the Microsoft Query dialog box).

If you accidentally close the Select Data Source dialog box before selecting a data source, you can open it again by clicking **New Query** from the **File** menu.



The name of the table in the database.

The name of the database. This can be hidden by using the SQL options in the Easysoft ODBC Settings dialog box (refer to "Settings" in the module "Microsoft ODBC Administrator").

7. Highlight the CURRENCY_RATE table and click **Add** - the table is added to the Microsoft Query Table pane (see next illustration).
8. Click the **Close** button on the Add Tables dialog box.

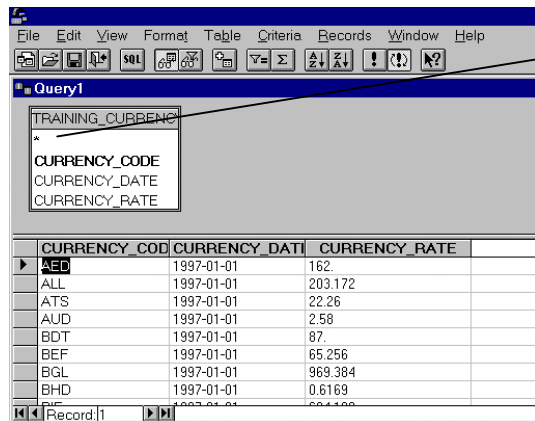


If you cannot see the Table pane (see next illustration) at all, ensure that the **Tables** option under the **View** menu is selected (i.e. there is a tick by it). If isn't selected, click on it to select it. The Table pane should now be visible, and it should contain the CURRENCY_RATE table.

If you can see a third pane between the Table pane and the Data pane, don't worry. This is the Criteria pane, which is explained later. If you want to hide it for now, de-select it from the **View** menu by clicking to remove the tick.

If you added the wrong table by mistake, then delete it by highlighting one of the fields in the table and then selecting **Delete** from the **Edit** menu.

9. To see the contents of a field, highlight the field using the left mouse button, and drag it into the data pane (or double-click on the field). To select all the columns, drag the asterisk into the data pane. Microsoft Query now looks like this.



Drag the asterisk into the Data pane to see all the data.

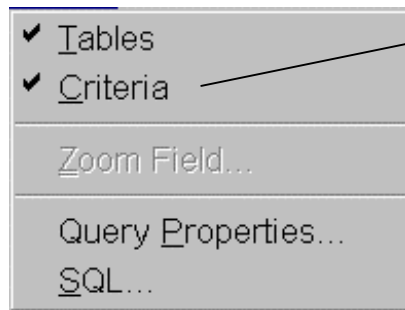
The Table pane.

The Data pane.

Add Criteria

At this stage, we've got all the currency rates known to the system. Modify the query to view data for just one currency (U.S. dollars).

1. Select the **View** menu, and toggle the **Criteria** option so that there is a tick by it - the Criteria pane should then become visible between the Table and Data panes.



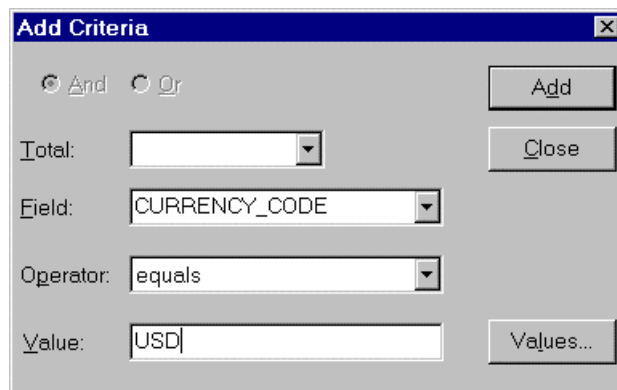
Criteria should be selected so that the Criteria pane is visible.

If this is selected, and the pane is not visible, then place the mouse over the division line between panes, and drag the bar downwards.



Alternatively, you can click the Show/Hide Criteria button to toggle the Criteria pane on and off.

2. Select **Add Criteria...** from the **Criteria** menu option. The Add Criteria dialog box appears (here shown with the first criterion entered).



Set the following criteria using the Add Criteria dialog box:

Field	Operator	Value
(= Column name)		
CURRENCY_CODE	equals	'USD'

Ignore the Total entry box.

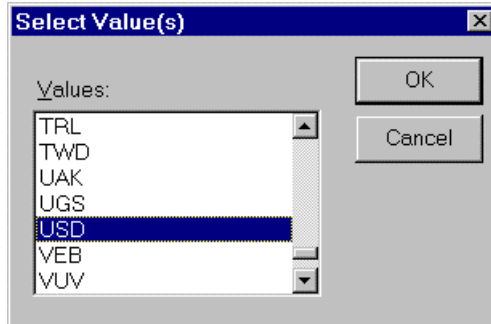
Field in the dialog box is equivalent to the Column name in the table.

Criterion is the criterion we want in the SQL that is generated. To generate this, enter the value shown for Operator. The "equals" operator will be converted to the "=" symbol in SQL.

- When you have entered the criterion for a field, click **Add**. The criterion that you entered becomes visible in the Criteria pane (see next diagram), and the Add Criteria dialog box is cleared.

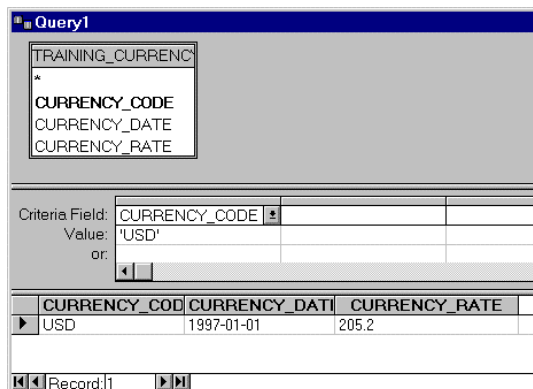


When it is possible to enter a value, you can either enter values directly in the Value field, or you can use the Select Value(s) dialog box.



In this example, we see that there are just five possible years to choose from. You needn't use this method of selecting a criterion value, but the advantage is that you can be sure that the value you choose exists in the database. For example, if you had entered a value of ~~USD~~ in the Add Criteria dialog box, no errors would have been generated (there are none), but you wouldn't have got any data back, because there are no currencies with a code of UDS. Another advantage is that you do not need to know whether data is character data or numeric - if it is character data, the required quote marks are automatically added.

- When you have set all the criteria, press the **Close** button on the **Add Criteria** dialog box. The data which matches the criteria is downloaded.

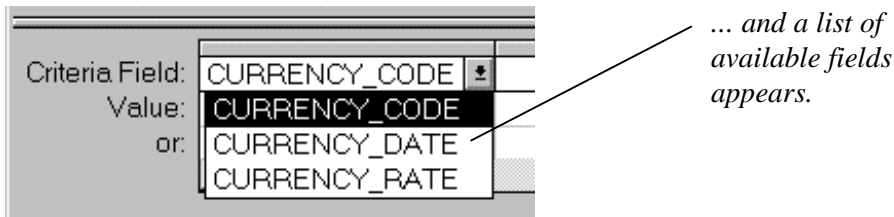
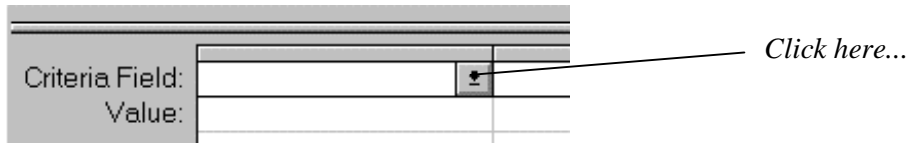


The Criteria pane.


Alternative Methods of Creating Criteria




1. Another method is to place the cursor in the Criteria field. A drop-down button appears. Click on this button to display a list of fields.

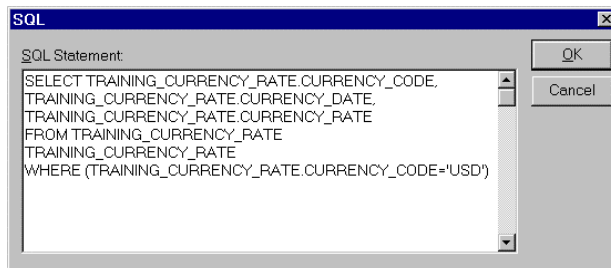


Click on the field for which you want to create a selection criterion (BUDGET_CODE in our case). The field name appears in the Criteria Field. Then type a value in the Value field; to see the currency rates for U.S. dollars, type 'USD' and then move the cursor to another field.

2. Another method of criteria selection is to highlight a data value that you want to set as a criterion and then click the Criteria Equals button (). (Obviously, there must be data in the data pane for you to do this).

View SQL

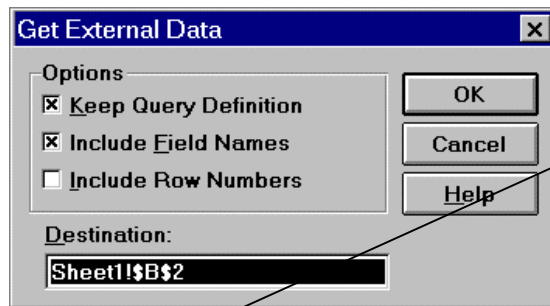
1. Look at the SQL that is sent to the database by the query. Press the SQL () button (or select **SQL...** from the **View** menu). The SQL dialog box appears showing the SQL statement that was just used in the query.



2. It is possible to modify the SQL directly and then re-send the query. For now, just press the **Cancel** button.

Return Data to Excel

1. To return data to Excel from Microsoft Query, select **Return Data to Microsoft Excel** from the **File** menu. The Get External Data dialog box appears.



Here the Destination list box contains the location of the first column in the first row on the data sheet in which the returned data will be placed.

2. The Keep Query Definition option and the Include Field Names option should be selected (i.e. cross in box).
3. The Destination value is the first cell of the first row in the spreadsheet where the data will be returned. Check that the value it contains is one that you want (if it doesn't, type it in the entry box).
4. Click **OK**. The data is returned to Excel.

	A	B	C	D
1				
2		CURRENCY_CODE	CURRENCY_DATE	CURRENCY_RATE
3		USD	1997-01-01 00:00:00.00	205.2
4				

You can now use Excel to create charts, reports etc.

Get External Data options

The following paragraphs explain the Options which are available on dialog box.

Keep Query Definition

This allows you to save the query definition in the *worksheet* (not the same as saving the query in a .QRY file) so that you can update the result set (i.e. the data that is returned to Excel) directly from Excel.

Include Field Names

If you want column headings to be shown, then select this option. The column headings are the field names that are used in the file from which the data is taken.

Include Row Numbers

Row numbers appear in the first column of the returned data; they are not part of the original data.

Destination:

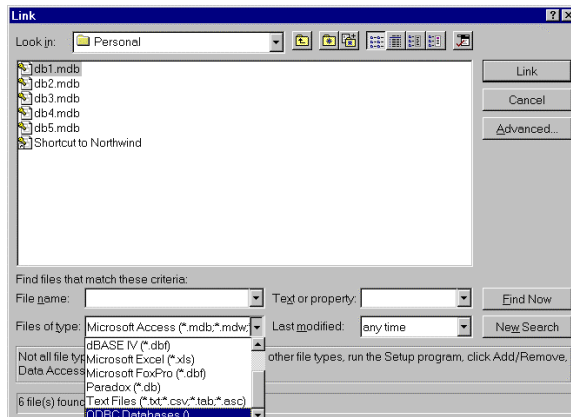
If the Destination: list box contains a range, then the data is returned to those cells specified by the range (hence not all of it may be displayed). If the list box references just one cell, then that cell is the location of the first column of the first row of the returned data. The default value of the list box is taken directly from the Excel worksheet from which the Get External Data dialog box was called. If the Destination is not what you want, change the values in the list box.

Microsoft Access

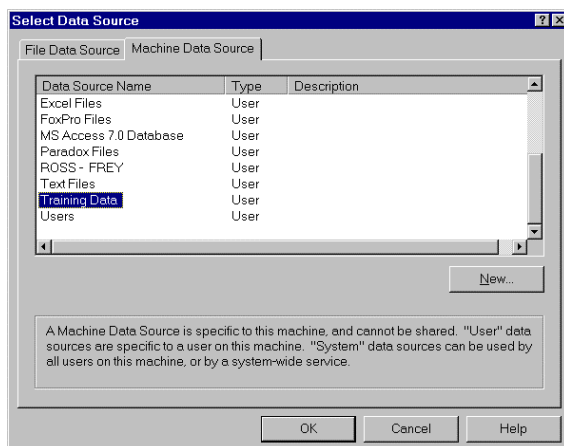
These Access examples are based on Microsoft Access version 7.0, for Windows 95. In these exercises you will download the entire contents of the CURRENCY_RATE table. Then you will set a criterion so that you see the rates for just one currency. Next, you will see how to view the SQL that is sent to the server. Finally, you will see how to view the indexes that are available on a table.

Download Entire Table

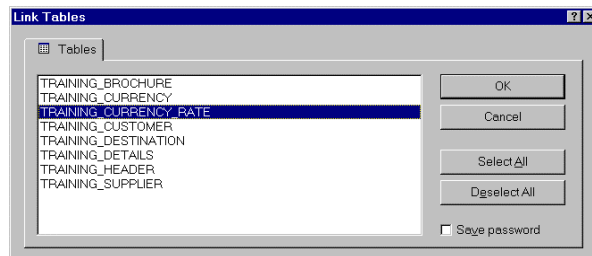
1. Start Microsoft Access and either open an existing database or create a new one.
2. From the **File** menu select **Get External Data** followed by **Link Tables...** The Link dialog box appears.



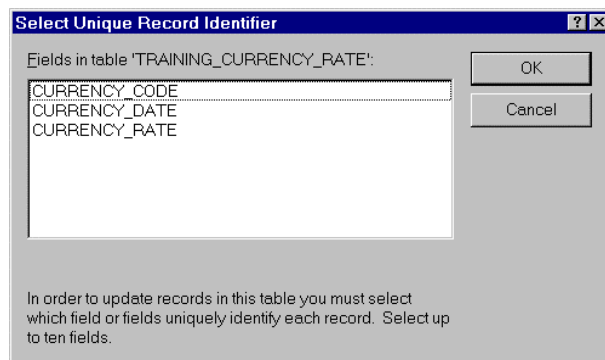
3. From the Files of type list box select **ODBC Databases()**. The Select Data Source dialog box appears if you have ODBC version 3.0 installed (with the Microsoft ODBC version 2.5 the SQL Data Sources dialog box appears).



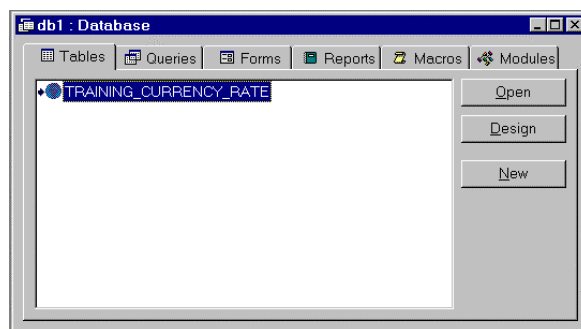
4. Select **TRAINING DATA** as the data source and click **OK** . (If the Easysoft ODBC Login Prompt dialog box appears, click **OK**). The Link Tables dialog box appears.



5. Select the **CURRENCY_RATE** table and click **OK**. The Select Unique Record Identifier dialog box appears.



6. To be able to update records you must define the fields which uniquely identify each record (in this case **CURRENCY_CODE** and **CURRENCY_DATE**). Since you will not be updating records, click **Cancel**. When the operation is complete the Database dialog box contains a list of linked tables - just one in this case.



7. Double clicking on a table displays the data. You can select subsets of the data and add and delete records. The changes you make are updated to the file on the Server which relates to the table you are changing. Double click the **CURRENCY_RATE** table to see server data in Microsoft Access.

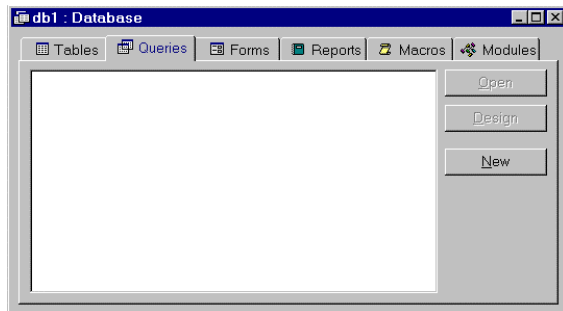
TRAINING_CURRENCY_RATE : Table		
CURRENCY_CODE	CURRENCY_DATE	CURRENCY_RATE
AED	01-Jan-97	162
ALL	01-Jan-97	203.172
ATS	01-Jan-97	22.26
AUD	01-Jan-97	2.58
BDT	01-Jan-97	87
BEF	01-Jan-97	65.256
BGL	01-Jan-97	969.384
BHD	01-Jan-97	0.6169
RIF	01-Jan-97	674.108

Record: 1 of 97

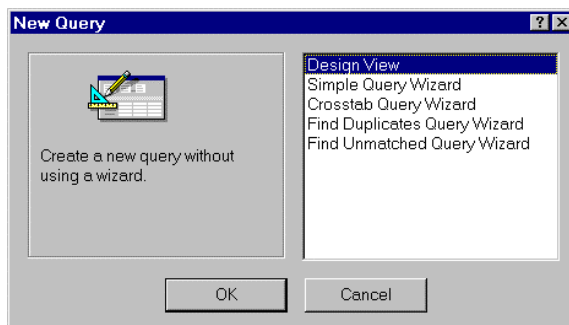
Add Criteria

In this exercise you will select records for a single year from the CURRENCY_RATE table.

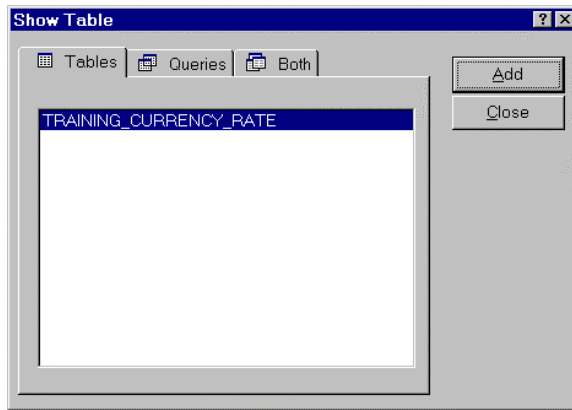
1. Click the **Queries** tab on the Database dialog box.



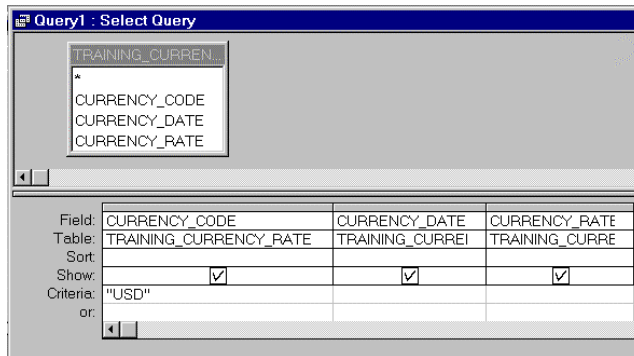
2. Click the **New** button. The New Query dialog box appears.




3. Ensure **Design View** is selected, then click the **OK** button. In the background you can see Microsoft Query, and in the foreground the Show Table dialog box appears. Ensure the CURRENCY_RATE table is highlighted in the Tables tab, and click the **Add** button.



4. The CURRENCY_RATE table appears in the Table pane of Microsoft Query. Click the **Close** button on the Show table dialog box. The Select Query dialog box is now visible (shown completed).

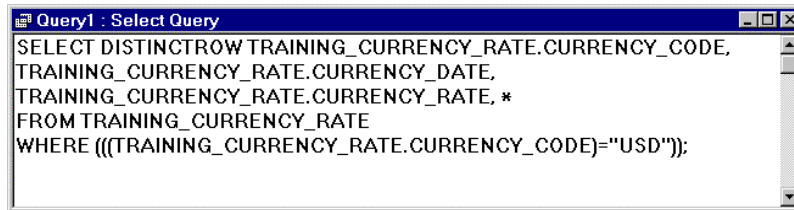


5. Select all the columns from the CURRENCY_RATE table in the Data pane, and drag them into the Query pane.
6. To select the data for U.S. dollars, enter the value 'USD' in the Criteria row for the CURRENCY_CODE field.
7. Select **Run** from the Query menu on the Access menu bar, or click the Run icon () on the toolbar.

The currency rate data is downloaded.

View SQL

1. To see the SQL that is sent, select **SQL** from the **View** menu.



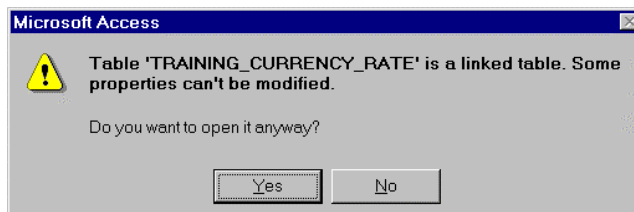
```

SELECT DISTINCTROW TRAINING_CURRENCY_RATE.CURRENCY_CODE,
TRAINING_CURRENCY_RATE.CURRENCY_DATE,
TRAINING_CURRENCY_RATE.CURRENCY_RATE, *
FROM TRAINING_CURRENCY_RATE
WHERE (((TRAINING_CURRENCY_RATE.CURRENCY_CODE)='USD'));

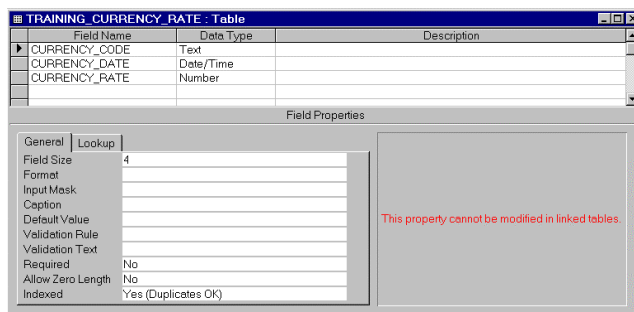
```

View Indexes

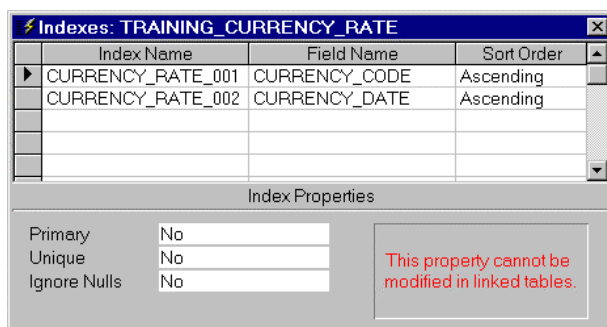
1. To view the indexes that are available on a table, select Design mode for the table (e.g. **Tables, Design** or **View, Table Design**). You will see the following message:



2. Click the **Yes** button; the table is displayed in Design mode.



3. Select **Indexes** from the **View** menu. The details of the available indexes for the table are displayed.



Lotus 1-2-3

These exercises are based on Lotus 1-2-3 release 5. In these exercises you will first set up Lotus 1-2-3 prior to downloading the entire contents of the CURRENCY_RATE table. Then you will set a criterion so that you see the rates for just one currency. Finally, you will see how to view the SQL that is sent to the server.

Note: Lotus 1-2-3 works with 16-bit ODBC only. Therefore, before doing these exercises, set up a 16-bit data source. Call it "Training Data - 16 bit".

Configure Lotus 1-2-3 to work with Easysoft ODBC

Before you can use Easysoft ODBC with Lotus 1-2-3, you must add a line to the `lotus.bcf` file. Use a text editor, or if you use a word-processor, ensure that you save the document as a text file.

1. Edit the **lotus.bcf** file, which will probably be in the default directory `\windows\lotusapp\dataens`.

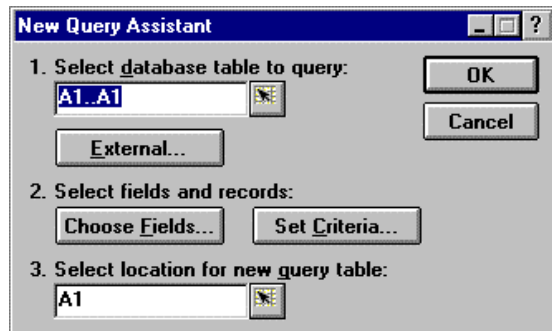
If the file is not in this directory, search for its location using the Windows Explorer by selecting the following sequence: **Tools, Find, Files or Folders**.

2. Add a line to the bottom of the **lotus.bcf** file containing the following:
`DN="ODBC" DD="ODBC" DL="DLODBC";`

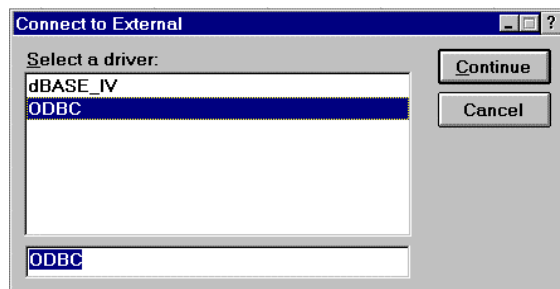
If this line already exists in the file, there is no need to repeat it.

Download Entire Table

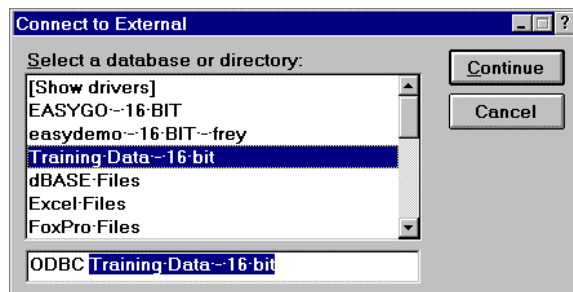
1. Ensure that you have set up a 16-bit data source. Then start Lotus and create a new worksheet if necessary.
2. Select the **Tools** menu, and from this select **Database**, followed by **New Query...** The New Query Assistant dialog box appears.



3. Click **External...** The Connect to External dialog box appears.



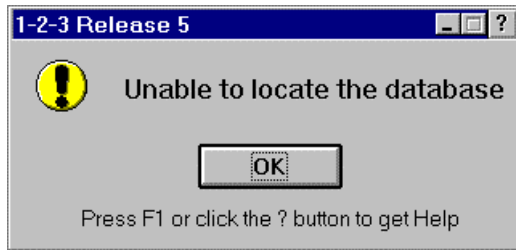
4. Highlight ODBC, then click **Continue**. A list of data sources appears.



Highlight the data source you want to use, and then click the **Continue** button.

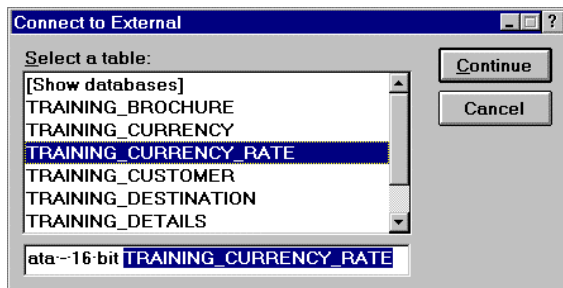


If you get the following message, you probably didn't connect to the 16-bit data source.



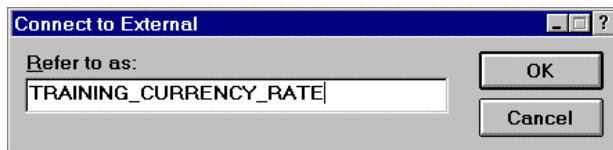
Ensure that you created it and that you connected to it.

5. A list of tables for each of the available databases appears.



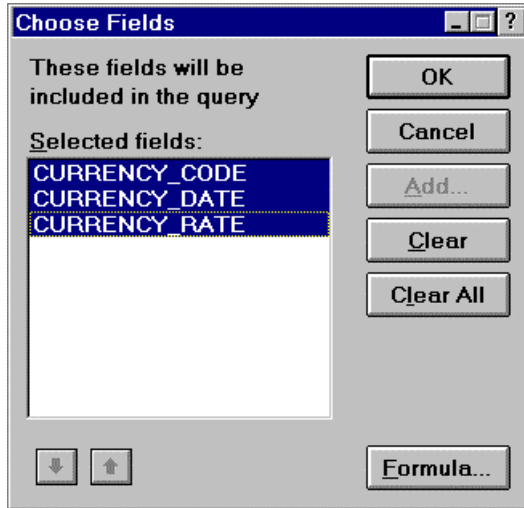
Highlight the CURRENCY_RATE table, and click the **Continue** button.

6. The Connect to External - Refer to as: option appears. Click **OK**.



7. The New Query Assistant re-appears. Click **Choose Fields...**

- The Choose Fields dialog box appears.



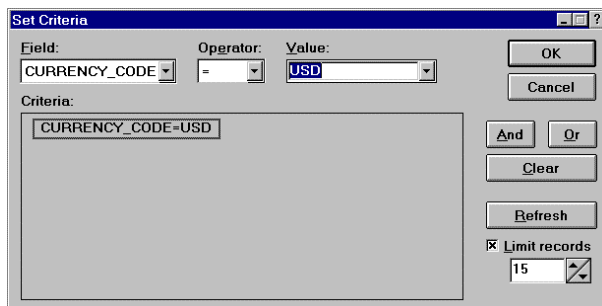
Highlight all the fields and click **OK**.

- The New Query Assistant re-appears. Click **OK** to download the data.

	A	B	C
1	CURRENCY_CODE	CURRENCY_DATE	CURRENCY_RATE
2	AED	01/01/97	162
3	ALL	01/01/97	203.172
4	ATS	01/01/97	22.26
5	AUD	01/01/97	2.58
6	BDT	01/01/97	87
7	BEF	01/01/97	65.256

Add Criteria

- Place the cursor in the downloaded data.
- The Query menu appears on the toolbar. From this, select **Set Criteria...** The Set Criteria dialog box appears.

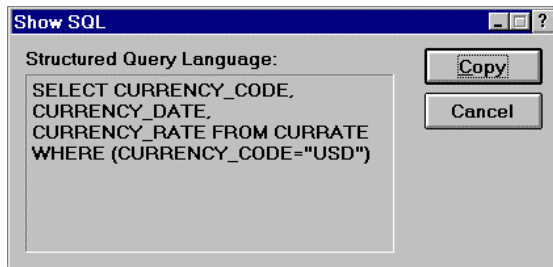


3. Click the **Clear** button to remove any default criteria.
4. Select the field, comparison operator and value from the three list boxes. The criteria you set appears in the Criteria window. Click **OK**.
5. The data in the spreadsheet changes to reflect the new criteria.

View SQL

This option is limited. You cannot change the SQL, and you may not even be able to view it in its entirety in Lotus 1-2-3.

1. Place the cursor in the downloaded data.
2. The Query menu appears on the toolbar. From this, select **Show SQL...** The Show SQL dialog box appears.



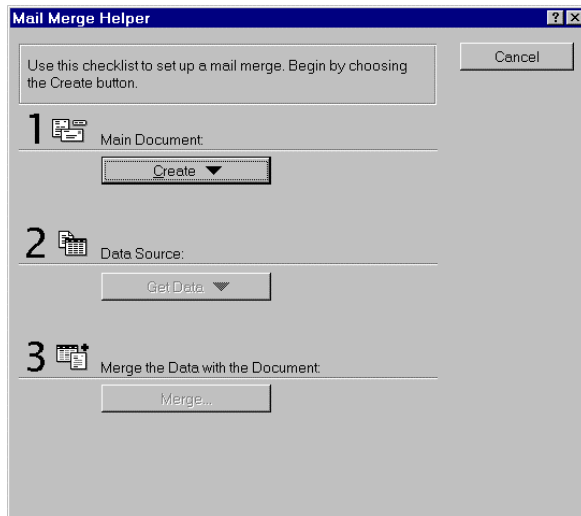
3. There are no scroll bars in this dialog box, so you may not be able to see all the SQL. However, you can copy it to the clipboard by clicking the **Copy** button. Then paste it into a text editor or word processor to view it. When you click either the **Copy** button or the **Cancel** button, the dialog box closes.

Microsoft Word Mail Merge

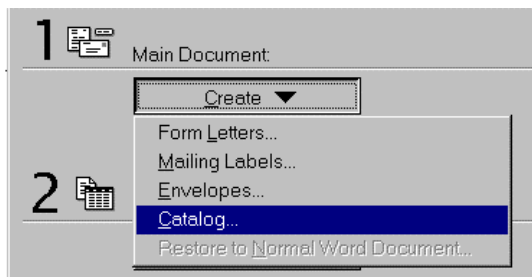
This example is based on Microsoft Word 7.0 for Windows 95.

Mail merge allows you to create documents which contain information imported from a data source. This example uses the currency data; you will see how to create a list of all currency rates.

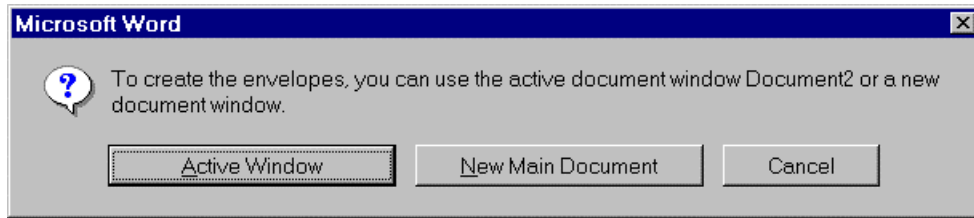
1. Open a new Word document.
2. Select **Tools, Mail Merge...** The Mail Merge Helper appears.



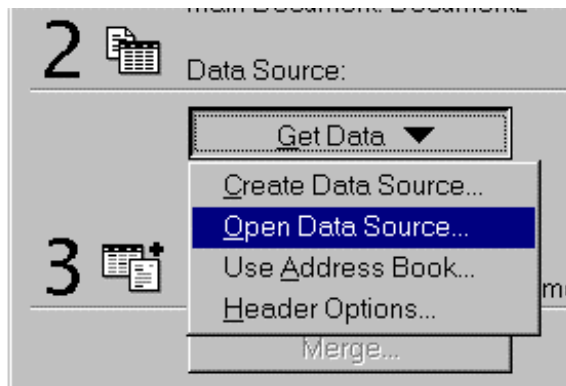
3. Create a document for the merged data - click the **Create** button and select the Catalog option



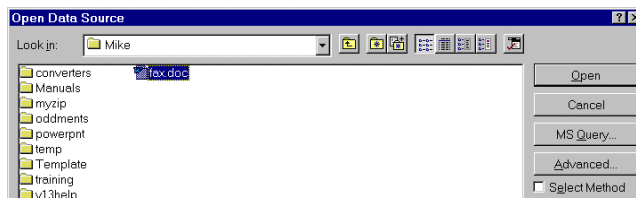
4. You are asked whether you want to use the active window or a new one. Select **Active Window** (you already created a new document in step 1 of this exercise).



5. Step two of the Mail Merge Helper is now available for use. Click the **Get Data** button

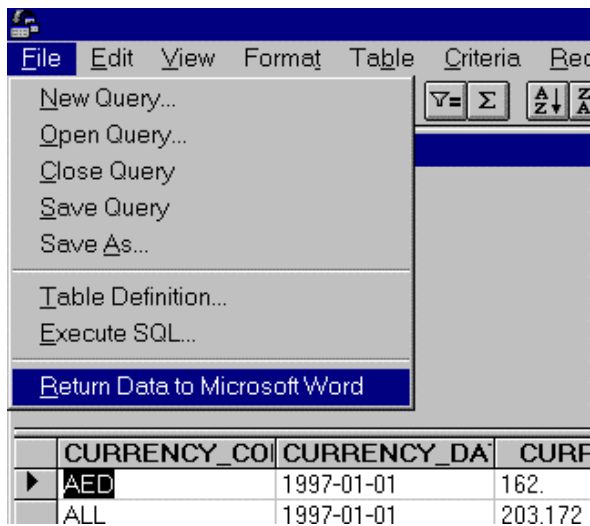


6. Select the **Open Data Source...** option. The Open Data Source dialog box appears.

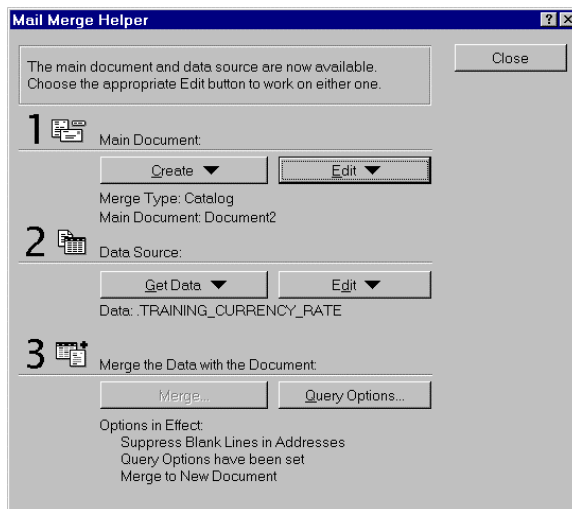


7. Click the **MS Query...** button. This starts Microsoft Query. You have already seen how to use this, so the details are not repeated. If you need to refresh your memory, look at Microsoft Excel at the start of this module.
8. Select the Training Data data source.
9. Add the CURRENCY_RATE table to the query.
10. Insert all the fields into the Data Pane.

11. To return the data to Word, select the **Return Data to Microsoft Word** option from the **File** menu.



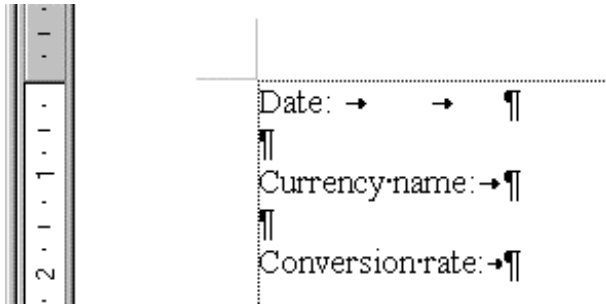
12. You will notice that the Merge option of Step 3 is not available. Click the **Close** button on the Mail Merge Helper.



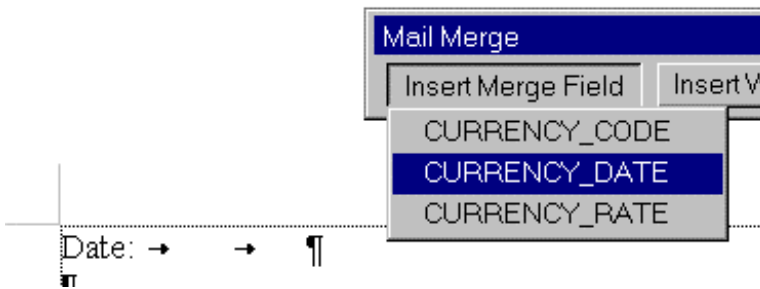
13. The Mail Merge toolbar is now available.



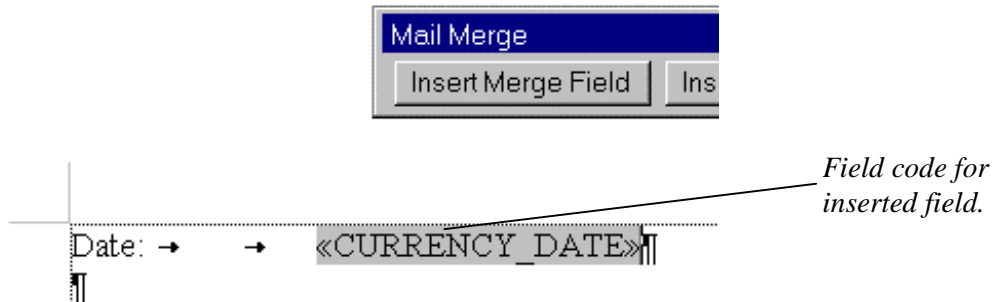
14. Type some text in the word document. Place the cursor where you want the merged data to appear. For example:



15. Click the **Insert Merge Field** option on the Mail Merge toolbar and select the field you want to insert.

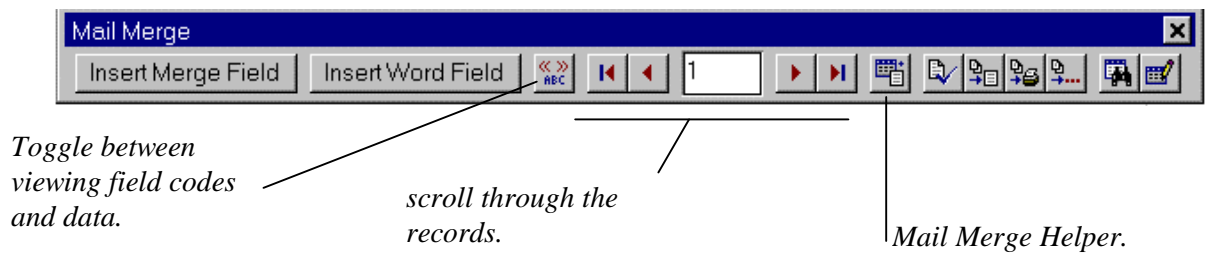


16. The field is inserted into the document.



17. Repeat this process for as many of the fields as you require.

18. Use the Mail Merge toolbar to view the data.

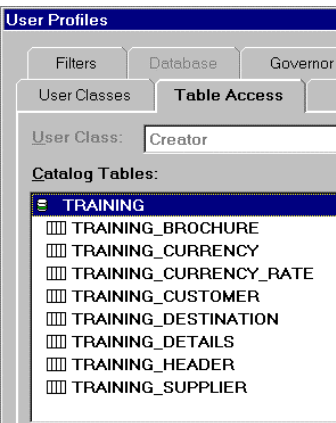


Note: to set criteria and view the SQL, use the Query Options button on the Mail Merge Helper to gain access to Microsoft Query.

Impromptu

These examples are based on Cognos Impromptu version 3.5. In these exercises you will create an Impromptu catalog as a preparatory step. Then you will download the entire contents of the CURRENCY_RATE table. Next, you will set a criterion so that you see the rates for just one currency. Finally, you will see the SQL that is sent to the server.

Note on names



The table and column names that you see on the Impromptu screen are not the same as the names defined in the Easysoft catalog. There are two differences:

- names are lower case
- the space character, rather than the hyphen is used as a separator

However, these are superficial changes; the underlying catalog tables and columns are name exactly as they are in the database, as indicated in the screen shot on the left.

Preparation: Create an Impromptu Catalog

Before you download data, you need an Impromptu catalog. This is a file which contains the information necessary for Impromptu to access and retrieve data from a database. In a similar manner to the Easysoft catalog, it doesn't contain data; it tells Impromptu how to get the data.

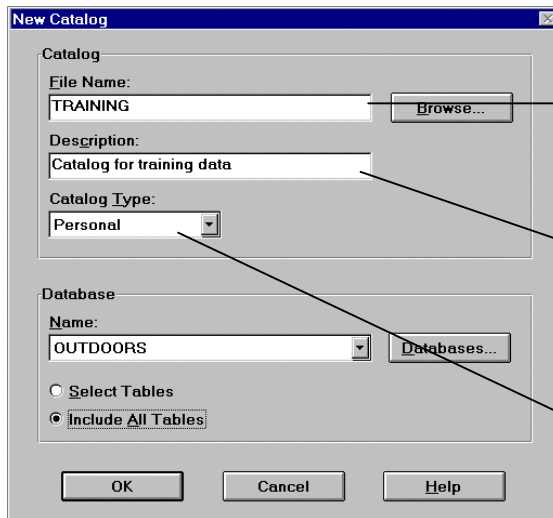
You can have many catalogs in Impromptu. Here you will create one which is used for the Easysoft training data.

1. Start Impromptu. On Windows, the default menu location under the **Start** menu is: **Programs, Cognos, Impromptu Administrator.**

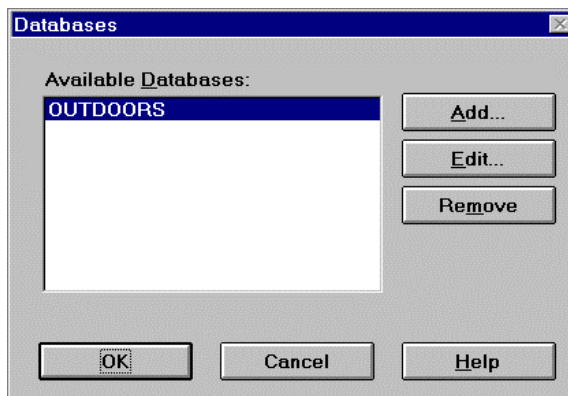
2. Select **Catalog, New...**



3. The New Catalog dialog box appears. When it is finally completed, it should look like this. For now, complete the Catalog section as follows:



4. The TRAINING database is not yet available. Click the **Databases...** button. The Databases dialog box appears.



5. Click the **Add...** button. The Database Definition dialog box appears. Complete it as follows:

The screenshot shows the 'Database Definition' dialog box with the following fields and options:

- Logical Database Name:** TRAINING
- Database Type/Gateway:** ODBC Gateway
- ODBC Data Source:** Training Data - 16 bit
- ODBC Connect String:** (empty)
- User Prompts:**
 - User ID
 - Password
- Timeouts:**
 - Connect: 0
 - Reply: 0
- Use Asynchronous Open

6. Click the **OK** button. Control passes back to the Databases dialog box. The TRAINING database now exists. Select it, and click the **OK** button.

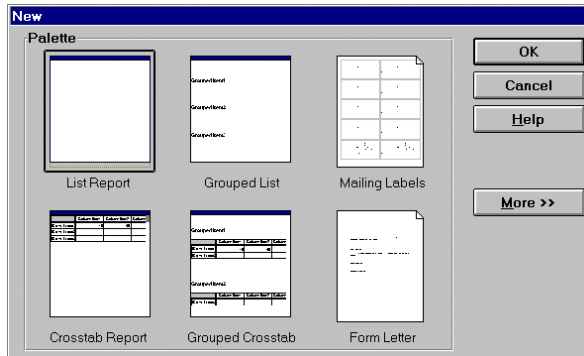
The screenshot shows the 'Databases' dialog box with the following content:

- Available Databases:**
 - OUTDOORS
 - TRAINING** (selected)

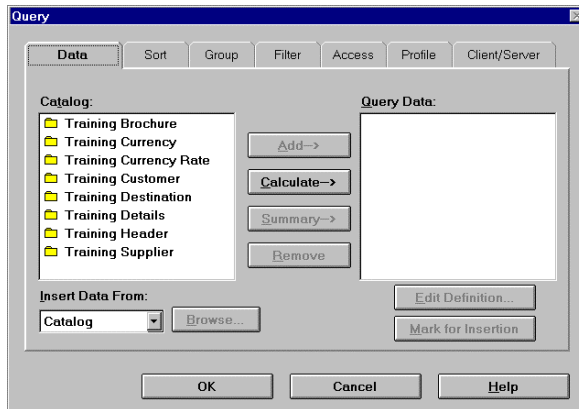
7. Control passes back to the New Catalog dialog box. Select the **Include All Tables** check box. The dialog box should now appear as shown in step 3. Click the **OK** button. Impromptu connects to the catalog (this may take a few seconds), and you should now be able to access the training data.

Download Entire Table

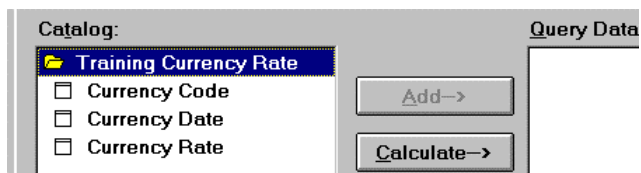
1. Select **File, New** from the menu bar. The New dialog box appears.



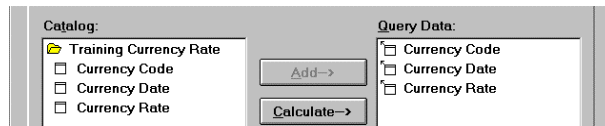
2. Highlight **List Report** (the default). Click **OK**. The Query dialog box appears.



3. Double-click the **Currency Rate** table in the Catalog list box. The columns in the table are shown.



- Highlight all three columns. Then click the **Add→** button. The columns are added to the Query Data list box.



- Click **OK**. The data is downloaded and appears in the report.

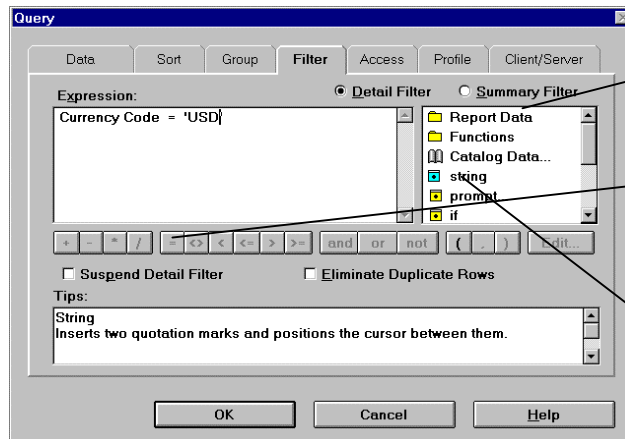
Currency Code	Currency Date	Currency Rate
AED	01-Jan-97	162
ALL	01-Jan-97	203.172
ATS	01-Jan-97	22.26
AUD	01-Jan-97	2.58
BDT	01-Jan-97	87
BEF	01-Jan-97	65.256
BGL	01-Jan-97	969.384
BHD	01-Jan-97	0.6169
BIF	01-Jan-97	624.108
BRL	01-Jan-97	2.112
BWP	01-Jan-97	7.404
CAD	01-Jan-97	2.808
CHF	01-Jan-97	2.718

Add Criteria

At this stage, we've got all the currency rates known to the system. Modify the query to view data for just one currency (U.S. dollars).

- Select the **Report, Query...** menu options. The Query dialog box appears.

- Click the **Filter** tab. You must use the expression builder to create the criterion, you cannot simply type the criterion in the Expression entry box. When you have finished, the dialog box should look like this.



1. Double click Report Data. Columns in the table are shown. Double-click Currency Code

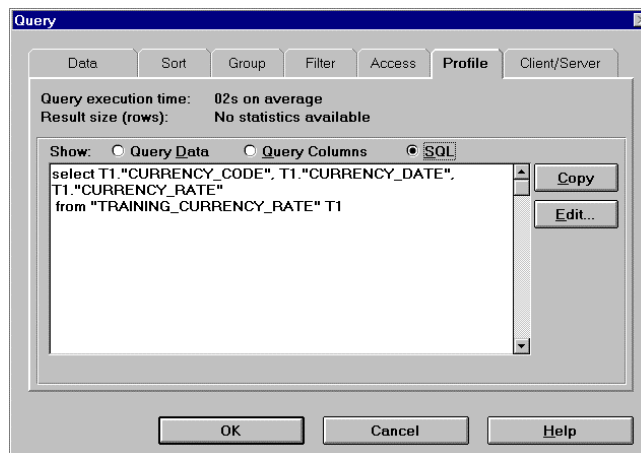
2. Click the equals button.

3. Double click string. Two single quote marks are placed in the Expression box. Place the cursor between them, and enter USD.

- Click **OK**. The currency rates for USD will be downloaded.

View SQL

- Look at the SQL that is sent to the database by the query. Select the **Report, Query...** menu options. The Query dialog box appears.
- Click the **Profile** tab.
- Select the **SQL** option. The SQL is shown in the window.



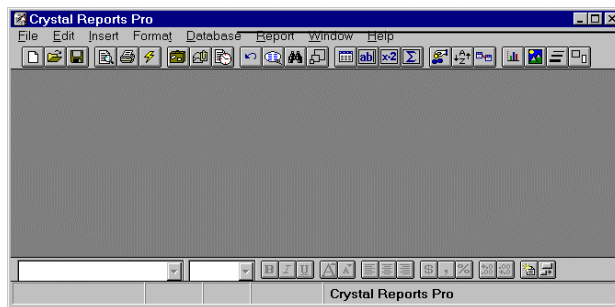
- It is possible to edit the query; for now, click **OK**.

Crystal Reports

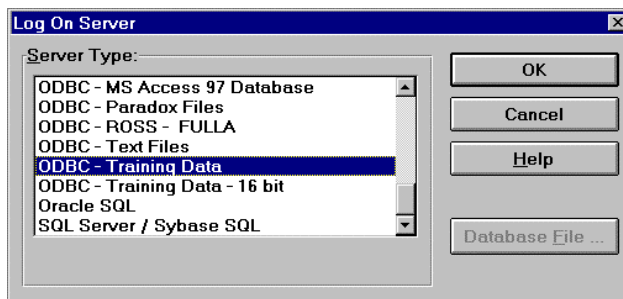
This example is based on Crystal Reports version 4.0. In these exercises you will download the entire contents of the CURRENCY_RATE table. Then you will set a criterion so that you see the rates for just one currency. Finally, you will see how to view the SQL that is sent to the server.

Download Entire Table

1. Start Crystal Reports. The main dialog box appears.



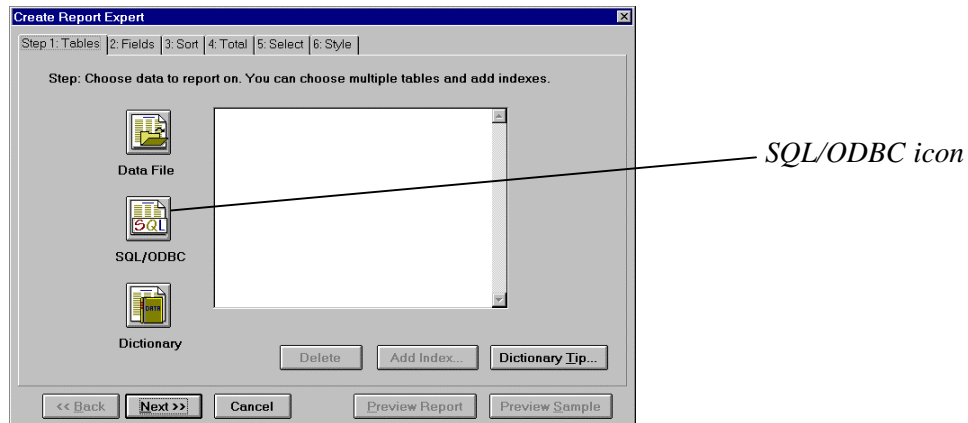
2. Select **Log On Server...** from the **Database** menu. The Log On Server dialog box appears.



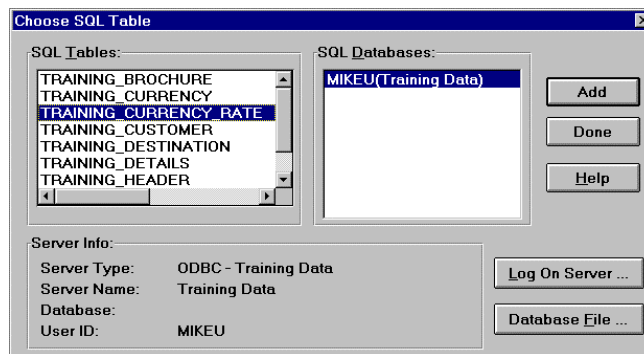
3. Select **ODBC - Training Data** from the Server Type list box and click **OK**.
4. If the Easysoft ODBC Login Prompt appears ensure that your server and catalog usernames and passwords are entered, and then click **OK**.
5. The Log On SQL Server dialog box appears. Click **OK**.



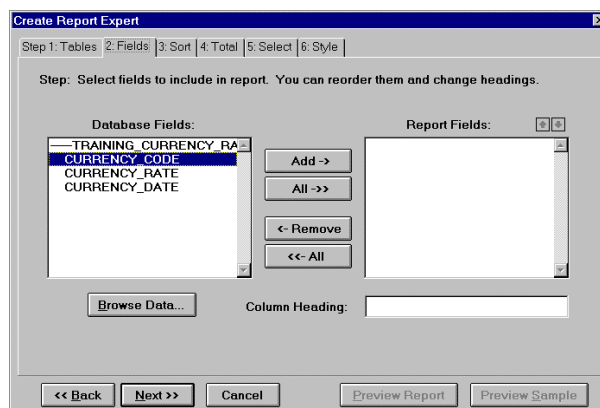
6. Control passes back to the main dialog box. Select the menu sequence: **File, New, Standard Expert...** Step 1 of the Create Report Expert dialog box appears.



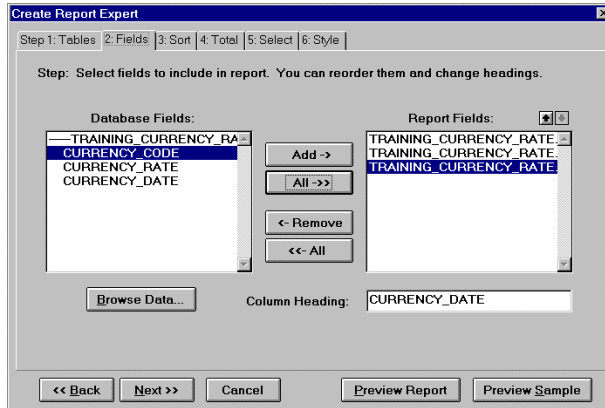
7. Click the **SQL/ODBC** icon. The Choose SQL Table dialog box appears.



8. Select the **TRAINING_CURRENCY_RATE** table and click **Add**.
9. You will not see any change in the Choose SQL Table dialog box, but the table appears in the list in the Create Report Expert dialog box. Click **Done** on the Choose SQL Table dialog box.
10. Click the **Next>>** button on the Create Report Expert dialog box. Step 2 of the Expert appears.



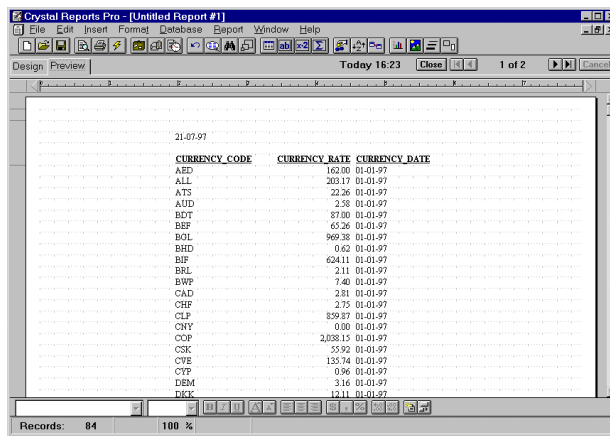
11. Click the **All->>** button to add all the columns to the report.



12. Click the **Preview Sample** button. The Preview Sample dialog box appears.



13. Since we know that there is not a huge amount of data, click **OK** to accept the default option to view all records.

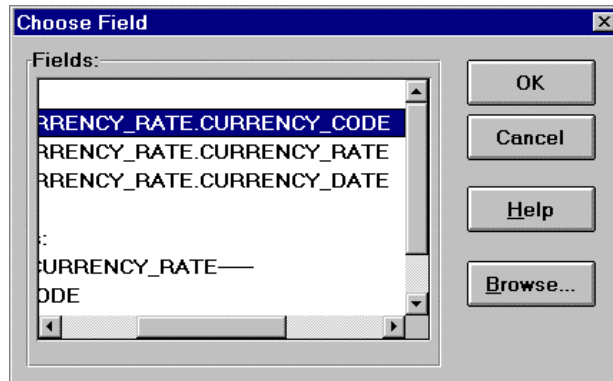


Add Criteria

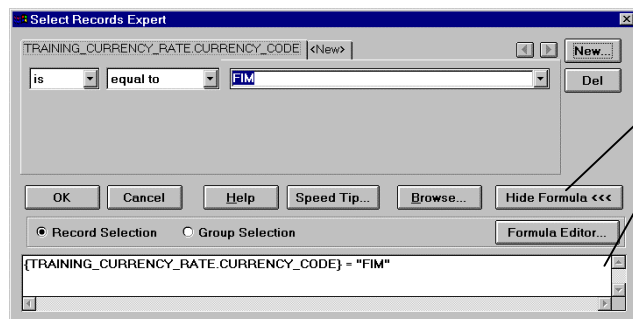
At this stage you've got all the currency rates known to the system. Modify the query to view data for just one currency (Finnish Markka, FIM).

There is more than one route to setting criteria in Crystal Reports. You may prefer a different method to the one shown here.

1. Choose the **Select Records Expert** option from the **Report** menu. The Choose Field dialog box appears.



2. Select **CURRENCY_CODE** (either from the Report Fields section or the Database Fields section) and click **OK**. The Select Records Expert dialog box appears.

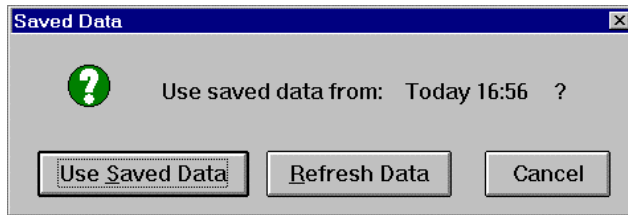


Use this button to show and hide this part of the dialog box.

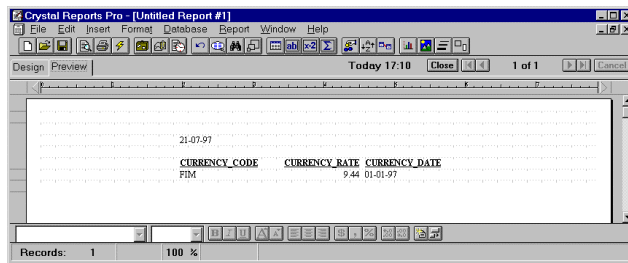
This formula is not the SQL that is sent to the server.

3. Set the following criterion using the dropdown boxes at the top of the dialog box.
 First box **is**
 Second box: **equal to**
 Third box: **FIM**

- Click the **OK** button. You will be asked whether you want to use the saved data or refresh the data. Select **Use Saved Data**, since you know that the data on the server has not changed.

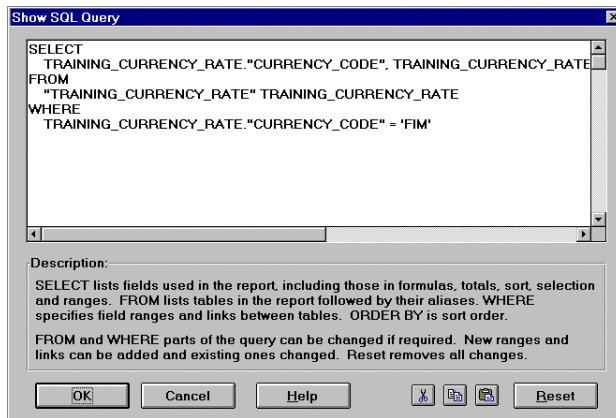


- The result set shows the data for FIM.



View SQL

- Look at the SQL that is sent to the database by the query. Select the **Show SQL Query...** option from the **Database** menu. The Show SQL Query dialog box appears.



- It is possible to edit the query; for now, click the **OK** button to return to the main window.

7. RMS Administration

This module is based on the assumption that you know the basics of using the OpenVMS operating system with the DCL command language.

In this module you will learn how to

- ✦ determine the file organisation of an existing RMS data file using the DCL DIRECTORY command and the Analyze/RMS_File utility
- ✦ determine the indexes that are defined on an RMS data file using the Analyze/RMS_File utility in order to optimise the SQL queries which access that data file
- ✦ create a new FDL file using the Edit/FDL utility
- ✦ create a new empty data file using the specification contained in an FDL file
- ✦ modify the organisation of an existing data file
- ✦ add a key to a data file
- ✦ view and set file protection and use access control lists (ACLs)

At the end of the module there are some review questions and exercises.

Contents

Introduction	7-2
Determine File and Record Structure	7-4
Method 1 - Directory Command	7-4
Method 2 - Use FDL	7-5
FDL for CURRENCY_RATE.DAT	7-6
Determining Fields in a Record	7-7
Determine the Keys	7-8
Creating and Maintaining Files	7-9
Define a New FDL File	7-9
Tutorial: Create a New RMS Data File	7-18
Add a Further Key to an Indexed File	7-19
Summary	7-20
Review Questions	7-21
Exercises	7-22
Review Answers	7-24
Answers to Exercises	7-25
Supplement: FDL File Structure	7-26
Description of KEY Secondary Attributes	7-27
Supplement: File Protection	7-30

Introduction

RMS provides three basic file organisations, namely Sequential, Relative and Indexed. This module deals with the following related issues:

- Before an RMS file can be accessed by an ODBC-compliant application, its record structure has to be mapped to a table structure. One of the functions of the Easysoft PC Administrator is to enable a user to enter this mapping information. In some cases, you may not know the structure of the files that you want to access using ODBC.
- Typically, indexes are defined on RMS data files; if you know what fields these indexes reference, you can write SQL queries that use these indexes; this can greatly improve the performance of the query.
- In some cases, there may be an RMS data file that you wish to add an index to.
- You may also want to define a new data file.

The File Definition Language Facility is one of the OpenVMS utilities. It includes the File Definition Language (FDL), the Create/FDL utility and the Edit/FDL utility.

File Definition Language (FDL) files define the attributes of RMS files. You can use FDL to specify RMS options and parameters that can be used when creating a file, such as file organisation and the record format. Although you can use FDL with sequential and relative RMS files, it is especially useful for defining RMS indexed files. Key definitions, area definitions, placement options, and bucket-size settings are typical of the information used to define indexed files that have optimal run-time performance.

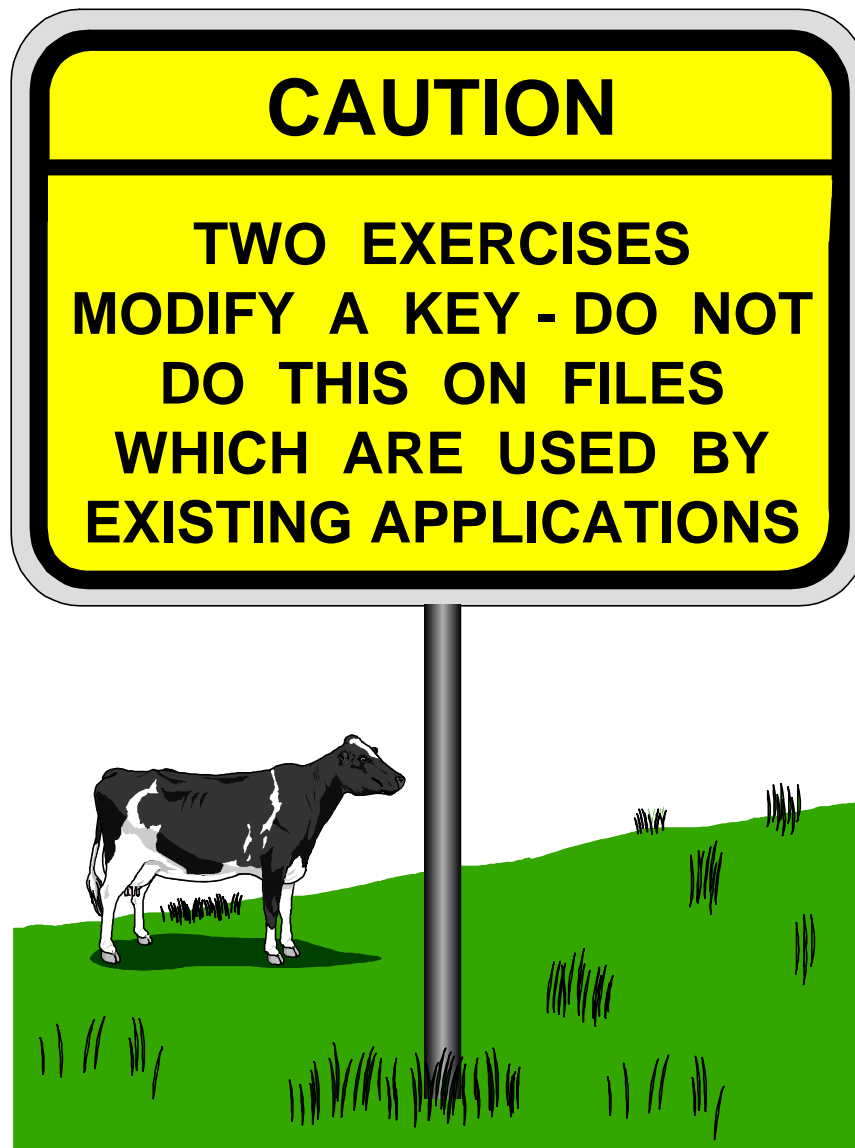
The following DCL commands invoke utilities that use FDL files:

- **ANALYZE/RMS_FILE** This is used to analyse the internal structure of an RMS file (see “Method 2 - Use FDL”, page 7-5).
- **EDIT/FDL** This invokes the FDL editor to create and modify File Definition Language files (see “Define a New FDL File”, page 7-9).
- **CREATE/FDL** This creates a new, empty data file using the specifications contained in an FDL file (see “Tutorial: Create a New RMS Data File”, page 7-18).
- **CONVERT** This is used to copy records from one file to another, changing the organisation and format of the input file to that of the output file (see “Note - Convert command”, page 7-20).

Throughout this module we will base the examples on an RMS data file called CURRENCY_RATE.DAT. The following logicals have been set up:

EASYSOFT_SQL_DATA points to the directory containing the training data.
EASYSOFT_SQL_FDL points to the directory containing FDLs for the training data.
EASYSOFT_SQL_CATALOG points to the catalog directory.

Sometimes, users are unable to access files using Easysoft ODBC for RMS because they do not have the correct privileges to read and/or write to the file. At the end of this module there is a brief overview of how to set file protection privileges, and how to use access control lists so that you can allow users access to the files you wish them to use.



Determine File and Record Structure

The CURRENCY_RATE.DAT file is an indexed file with a fixed record length of 23 bytes. The file contains data corresponding to three fields:

CURRENCY_CODE 4 BYTES	CURRENCY_RATE 8 BYTES	CURRENCY_DATE 11 BYTES
--------------------------	--------------------------	---------------------------

There are two methods of determining the file organisation.

1. Use the DIRECTORY/FULL command
2. Obtain details from an FDL file which can be generated from the data file

Method 1 - Directory Command

The easiest way to determine the file organisation of a file is to use the /FULL qualifier with the DIRECTORY command. The shaded lines below contain the required information.

```
$ DIRECTORY/FULL EASYSOFT_SQL_DATA:CURRENCY_RATE.DAT
Directory DKA300:[TRAIN_RMS.DATA]
CURRENCY_RATE.DAT;4          File ID: (15785,9,0)
Size: 45/45                 Owner: [CAROLYN]
Created: 10-JUL-1997 16:49:06.97
Revised: 11-JUL-1997 14:00:18.55 (67)
Expires: <None specified>
Backup: <No backup recorded>
Effective: <None specified>
Recording: <None specified>
File organization: Indexed, Prolog: 3, Using 1 key
                      In 2 areas
Shelved state: Online
File attributes: Allocation: 45, Extend: 6, Maximum bucket size: 3
                  Global buffer count: 0, No version limit
                  Contiguous best try
Record format: Fixed length 23 byte records
Record attributes: Carriage return carriage control
RMS attributes: None
Journaling enabled: None
File protection: System:RWED, Owner:RWED, Group:RE, World:
Access Cntrl List: None

Total of 1 file, 45/45 blocks.
$
```


Method 2 - Use FDL

The Analyze/RMS_File utility allows you to examine the structure of an RMS file. Various qualifiers are available; we will look at the /FDL qualifier.

Note

FDL (File Definition Language) files are described in more detail later (Creating and Maintaining Files, page 7-9). Basically, an FDL file is a description of a data file.

Various sections in the FDL file describe the structure of the data file, for example, there are sections called FILE, RECORD and KEY, which contain information about the file, the record structure and the keys respectively.

The /FDL qualifier generates an FDL file describing the OpenVMS RMS data file being analysed. By default, the /FDL qualifier creates a file with the file type .FDL and the same file name as the input data file. To assign a different type or name to the FDL file, use the /OUTPUT qualifier. If the data file is corrupted, the FDL file contains the Analyze/RMS_File utility error messages. You cannot use wildcards or multiple file specifications with the /FDL qualifier.

For indexed files, the FDL file contains special analysis sections you can use with the EDIT/FDL Optimize script to make better design decisions when you reorganise the file.

Assume that we want to determine the file and index structure of the CURRENCY_RATE.DAT file. The procedure for generating an FDL file is:

1. At the DCL command prompt type:

```
$ ANALYZE /RMS_FILE /FDL EASYSOFT_SQL_DATA:CURRENCY_RATE
```

Note. When using logicals as path names, a colon (:) must be placed between the logical path name and the file name.

The CURRENCY_RATE.FDL file is created in the current directory.

2. View the FDL file that is generated. At the DCL command prompt type:

```
$ TYPE CURRENCY_RATE.FDL
```

The complete content of the FDL file is shown on the next page.

FDL for CURRENCY_RATE.DAT

IDENT "11-JUL-1997 14:15:57 OpenVMS ANALYZE/RMS_FILE Utility"

IDENT shows date/time of creation, and the utility that created the FDL file.

SYSTEM

SYSTEM shows system identification information.

SOURCE OpenVMS

FILE

ALLOCATION 45
 BEST_TRY_CONTIGUOUS yes
 BUCKET_SIZE 3
 CLUSTER_SIZE 9
 CONTIGUOUS no
 EXTENSION 6
 FILE_MONITORING no
 GLOBAL_BUFFER_COUNT 0
 NAME

FILE section is used to specify file-related aspects of the data file. Described in detail below.

"DKA300: [TRAIN_RMS.DATA]CURRENCY_RATE.DAT;4"

ORGANIZATION indexed
 OWNER [CAROLYN]
 PROTECTION (system:RWED, owner:RWED, group:RE,

world:)

RECORD

BLOCK_SPAN yes
 CARRIAGE_CONTROL carriage_return
 FORMAT fixed
 SIZE 23

RECORD defines various controls for the record. Described in detail below.

AREA 0

ALLOCATION 36
 BEST_TRY_CONTIGUOUS yes
 BUCKET_SIZE 3
 EXTENSION 6

AREA relates to an RMS-specific region of an indexed file.

AREA 1

ALLOCATION 9
 BEST_TRY_CONTIGUOUS yes
 BUCKET_SIZE 3
 EXTENSION 3

KEY 0

CHANGES no
 DATA_KEY_COMPRESSION no
 DATA_RECORD_COMPRESSION yes
 DATA_AREA 0
 DATA_FILL 100
 DUPLICATES yes
 INDEX_AREA 1
 INDEX_COMPRESSION no
 INDEX_FILL 100
 LEVEL1_INDEX_AREA 1
 NAME "CURRENCY_CODE"
 NULL_KEY no
 PROLOG 3
 SEG0_LENGTH 4
 SEG0_POSITION 0
 TYPE string

Each key in an indexed file has a KEY section.

ANALYSIS_OF_AREA 0
 RECLAIMED_SPACE 0

ANALYSIS_OF_AREA section automatically generated by Analyze/RMSFile utility. Contains only one section - RECLAIMED_SPACE - which shows the number of blocks in the area which have been reclaimed by the Convert utility.

ANALYSIS_OF_AREA 1
 RECLAIMED_SPACE 0

ANALYSIS_OF_KEY	0
DATA_FILL	55
DATA_KEY_COMPRESSION	0
DATA_RECORD_COMPRESSION	-14
DATA_RECORD_COUNT	84
DATA_SPACE_OCCUPIED	12
DEPTH	1
INDEX_COMPRESSION	0
INDEX_FILL	2
INDEX_SPACE_OCCUPIED	3
LEVEL1_RECORD_COUNT	4
MEAN_DATA_LENGTH	23
MEAN_INDEX_LENGTH	6

ANALYSIS_OF_KEY section is automatically generated by the Analyze/RMSFile utility.

Refer to Supplement for details of each entry.

Determining Fields in a Record

A problem arises if you do not know the fields structure within an RMS file. There are no utilities which can help you determine this, since the concept of fields within a record is a semantic overlay on the physical structure of a record.

It may be possible to get an idea of the start position and end position of fields using the index definitions, but this breaks down if the index segments do not correspond to complete fields (for example, there could be an index that is defined on the first 5 characters of a 15 character string field).

You can obtain field information from the definitions contained in the applications that use the RMS data files.

Determine the Keys

The keys available on a file can be determined using an FDL. Each key of an indexed file has a KEY section. The primary key must be KEY 0. The value for secondary keys can be between 1 and 254.

This is the FDL definition for KEY 0 from the CURRENCY_RATE.FDL file:

```
KEY 0
  CHANGES                no
  DATA_KEY_COMPRESSION  no
  DATA_RECORD_COMPRESSION yes
  DATA_AREA              0
  DATA_FILL              100
  DUPLICATES              yes
  INDEX_AREA              1
  INDEX_COMPRESSION      no
  INDEX_FILL              100
  LEVEL1_INDEX_AREA      1
  NAME                    "CURRENCY_CODE"
  NULL_KEY                no
  PROLOG                  3
  SEG0_LENGTH             4
  SEG0_POSITION           0
  TYPE                    string
```

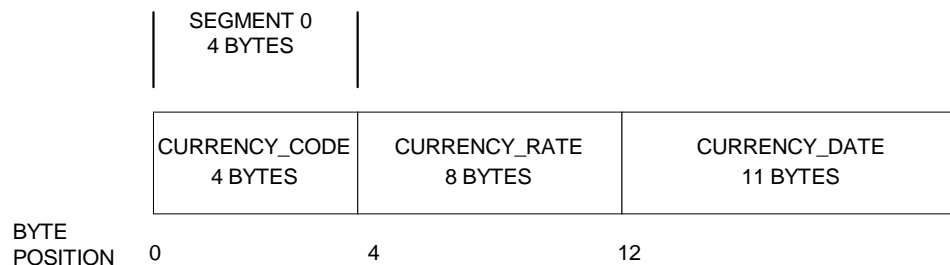
Each segment in a key has an entry.

LENGTH is the length of a segment and POSITION is the byte starting position.

For our purposes, the important attributes are SEGn_LENGTH and SEGn_POSITION.

An key can contain up to eight segments (that is, sections) in different parts of the record. SEGn_LENGTH refers to the length of each segment. SEGn_POSITION refers to the starting position (in bytes) of the segment. A segmented key (i.e. one with more than one segment) must be a STRING type.

The diagram below shows how the key in the CURRENCY_RATE record corresponds to the fields. The index is a primary key index (it is labelled KEY 0). It is defined on the CURRENCY_CODE field.



Creating and Maintaining Files

The Edit/FDL utility (also known as the FDL editor) helps you create, maintain and tune OpenVMS RMS files, particularly indexed files. Using the Edit/FDL utility, you can add, modify or delete file attributes explicitly or you can invoke scripts that ask questions which the Edit/FDL utility uses to make file attributes decisions.

The output from the Edit/FDL utility is an FDL file which is used as a model or template for the data file you wish to create. The file describes in ASCII text the attributes of the data file that it will be used to create (in this case, it will be an indexed file). There are several methods of creating FDL files but it is preferable to use the Edit/FDL utility because it enforces the FDL syntax.

FDL files are divided into logical sections labelled with a primary attribute such as SYSTEM, FILE, DATE, KEY, and so forth. Most primary attributes use secondary attributes to specify the file attributes for the resultant RMS file. FDL file structure is described in “Supplement: FDL File Structure”, page 7-26.

After you create an FDL file, you can create the RMS file modelled on the FDL file using the DCL commands CREATE/FDL and CONVERT/FDL (see “Tutorial: Create a New RMS Data File”, page 7-18).

Define a New FDL File

There are two parts to this section - first, there is a brief overview of the editor, then there is a step-by-step guide to defining an FDL file.

Editor Overview

You use the Edit/FDL utility by answering questions which control the operation of the editor, and which determine the values of variables which may be used either in calculating various file parameters or setting FDL attribute values.

Every question (with the exception of the Help-topic prompt you see while in the Help Function) that is asked has the same structure:

Question-text (answer-range)[default-answer] : your-answer

For some questions you must choose from several keyword options. These options are listed either in the accompanying menu (if present) or in an option list, which is enclosed by parentheses, such as below:

(Option1 Option2 Option3 ...)

Question-text (Keyword)[default-answer] : your-answer

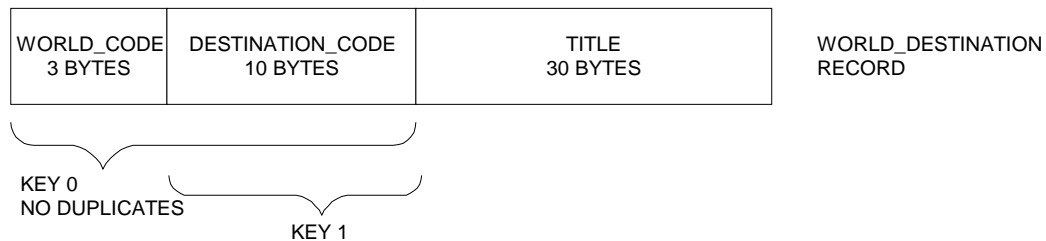
Ctrl/Z may be used to exit the Edit/FDL utility if you are currently at the main level. At lower levels, Ctrl/Z returns you to the main level.

You can use Ctrl/C to exit from the Edit/FDL utility at any time. Avoid using Ctrl/Y to abort the Edit/FDL utility because that may leave the scrolling region of a DEC_CRT terminal in an undetermined state. If this happens, use the DCL EXIT command to restore the terminal to its original state. This action is unnecessary if the next DCL command issued invokes a new image.

To get more information about any particular question, type "?" and press the Return key. When an invalid response is made to one of the FDL Editor's questions, the action taken is equivalent to the ? command.

Tutorial: Define WORLD_DESTINATION.FDL

In this exercise you will define an FDL which will be used to create an empty data file in the next exercise. Say that you want an FDL which can be used to create an indexed file WORLD_DESTINATION.DAT, which will later contain data corresponding to fields WORLD_CODE, DESTINATION_CODE and TITLE. (For a description of the WORLD_DESTINATION file refer to the module entitled "RMS Training Data").



There will be a primary key on the WORLD_CODE and DESTINATION_CODE combination and an alternate key on DESTINATION_CODE.

The three fields are all string fields.

The FDL you create will contain the following definitions:

```

FILE
  NAME                "EASYSOFT_SQL_DATA:WORLD_DESTINATION.DAT"
  ORGANIZATION        indexed

RECORD
  CARRIAGE_CONTROL    carriage_return
  FORMAT              fixed
  SIZE                43

KEY 0
  DUPLICATES          no
  SEG0_LENGTH          3
  SEG0_POSITION        0

KEY 1
  DUPLICATES          yes
  SEG0_LENGTH          10
  SEG0_POSITION        3

```

Perform these steps:

1. Change directory to EASYSOFT_SQL_FDL in preparation for creating an FDL:

```
$ SET DEF EASYSOFT_SQL_FDL
```

2. Start the FDL editor:

```
$ EDIT/FDL WORLD_DESTINATION
```

You do not need a file type unless you do not want to use the default (.FDL).

If you do not enter the file specification (WORLD_DESTINATION) at the command line, then you will be presented with the following prompt, at which you should enter the file specification:

```
_File: <file specification>
```

3. If a new file is being created, the following message appears. Press the **Return** key:

```
Parsing Definition File
```

```
DKA0: [TRAINER.RMS.FDL.TRAINEE]WORLD_DESTINATION.FDL; will be created.
```

```
Press RETURN to continue (^Z for Main Menu)
```

4. The top level of the FDL Editor appears. At the Main Editor Function prompt type **Invoke**. This will initiate a set of commands which you should follow.

OpenVMS FDL Editor

```

Add      to insert one line into the FDL definition
Delete   to remove one line from the FDL definition
Exit     to leave the FDL Editor after creating the FDL file
Help     to obtain information about the FDL Editor
Invoke   to initiate a script of related questions
Modify   to change an existing line in the FDL definition
Quit     to abort the FDL Editor with no FDL file creation
Set      to specify FDL Editor characteristics
View     to display the current FDL Definition

```

```
Main Editor Function      (Keyword)[Help] : INVOKE
```

5. The Script Title Selection menu appears. Enter **Indexed** at the prompt and press the Enter key.

```

Script Title Selection

Add_Key      modeling and addition of a new index's parameters
Delete_Key   removal of the highest index's parameters
Indexed      modeling of parameters for an entire Indexed file
Optimize     tuning of all indices' parameters using file statistics
Relative     selection of parameters for a Relative file
Sequential   selection of parameters for a Sequential file
Touchup     remodeling of parameters for a particular index

Editing Script Title      (Keyword) [ ] : INDEXED

```

(You can also define the FDL file using the Add menu options - however, this is a much longer process).

6. You are then prompted for the disk volume cluster size. Accept the default.

```

Target disk volume Cluster Size (1-2Giga) [3] :

```

7. You are then asked for the number of keys to define. Enter the value 2 at the prompt and press the Enter key.

```

Number of Keys to Define      (1-255) [1] : 2

```

8. You are then asked for the graph type for the first key (key 0). Accept the default value (**Line**).

```

Key 0 Graph Type Selection

Line   Bucket Size vs Index Depth      as a 2 dimensional plot
Fill   Bucket Size vs      Load Fill Percent      vs Index Depth
Key    Bucket Size vs      Key Length      vs Index Depth
Record Bucket Size vs      Record Size      vs Index Depth
Init   Bucket Size vs Initial Load Record Count vs Index Depth
Add    Bucket Size vs Additional Record Count vs Index Depth

Graph type to display      (Keyword) [Line] :

```

This option exists for historical reasons and was used to aid efficient disk usage.

9. You are then asked for the number of records that will be initially loaded into the data file. Enter the value zero here, because a new empty file will be created using the FDL.

```

Number of Records that will be Initially Loaded
into the File      (0-2Giga) [ ] : 0

```

10. You must then enter the number of records which will be added to the file after its creation. Enter the value 200.

```

Number of Additional Records to be Added After
the Initial File Load      (0-2Giga) [0] : 200

```


11. You are then asked whether records will generally be added in ascending primary key order. Accept the default value of **no**, because you do not know the order in which they will be added:

```
Will Additional Records Typically be Added in
Order by Ascending Primary Key (Yes/No)[No] : █
```

12. You are then asked whether records will generally be evenly distributed over the primary key values. Accept the default value of **no**.

```
Will Added Records be Distributed Evenly over the
Initial Range of Pri Key Values (Yes/No)[No] : █
```

13. Next, enter **Fixed** for the record format, because the size is constant:

```
(Fixed Variable)
Record Format (Keyword) [Var] : Fixed
```

14. Enter the value **43** for the record size

```
Record Size (1-32224)[-] : 43█
```

15. Enter the data type of the first key. In this case accept the default value: **string**. (The key is a combination of **WORLD_CODE** and **DESTINATION_CODE**, both of which are strings):

```
(Bin2 Bin4 Bin8 Int2 Int4 Int8 Decimal String Collated
 Dbin2 Dbin4 Dbin8 Dint2 Dint4 Dint8 Ddecimal Dstring Dcollated)
Key 0 Data Type (Keyword) [Str] :
```

16. The key is not segmented. Even though it contains a combination of two fields, these are contiguous; accept the default value of **no**:

```
Key 0 Segmentation desired (Yes/No) [No] :
```

17. Enter the length of the key i.e. number of bytes (**13**).

```
Key 0 Length (1-43)[-] : 13█
```

18. Enter the starting position of the key (**0**), which is beginning of the file in this case:

```
Key 0 Position (0-30) [0] : 0
```

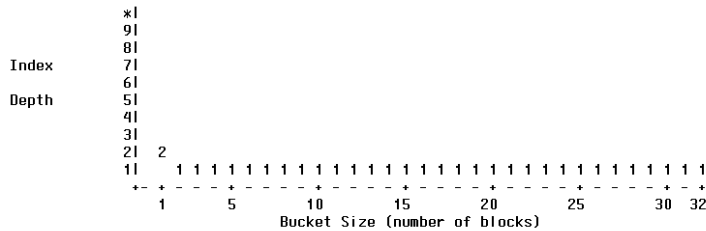
19. In this case duplicate key values are not allowed; accept the default.

```
Key 0 Duplicates allowed (Yes/No) [No] :
```

20. Accept the default values for the following four questions.

```
File Prolog Version          (0-3) [3]      :
Data Key Compression desired (Yes/No) [Yes]  :
Data Record Compression desired (Yes/No) [Yes] :
Index Compression desired    (Yes/No) [Yes]  : █
```

21. You will then see a graph similar to this



```
PV-Prolog Version      3 KT-Key  0 Type      String EM-Emphasis Flatter ( 3)
DK-Dup Key  0 Values   No KL-Key  0 Length    13 KP-Key  0 Position   0
RC-Data Record Comp  0% KC-Data Key Comp    0% IC-Index Record Comp  0%
BF-Bucket Fill      100% RF-Record Format    Fixed RS-Record Size   43
LM-Load Method  RMS_Puts IL-Initial Load    0 AR-Added Records    200
(Type "FD" to Finish Design)
Which File Parameter  (Mnemonic)[refresh]   : FD █
```

22. Enter the value **FD** (Finish Design) to complete the design. You will then be asked a further series of questions.

23. Enter a title for the FDL. For example, **WORLD_DESTINATION**.

```
Text for FDL Title Section (1-126 chars)[null]
: WORLD_DESTINATION
```

24. Enter a file specification for the data file that will be created.

```
Data File file-spec (1-126 chars)[null]
: EASYSOFT_SQL_DATA:WORLD_DESTINATION.DAT
```

25. Accept the default value (Carriage-Return) for the record delimiter.

```
(Carriage_Return FORTRAN None)
Carriage Control (Keyword) [Carr] :
```

26. Information about the index is then shown. Accept the suggested default value for the bucket size for the key:

```

Emphasis Used In Defining Default:      (   Flatter_files   )
Suggested Bucket Sizes:                 (   3   3   12 )
Number of Levels in Index:               (   1   1   1 )
Number of Buckets in Index:              (   1   1   1 )
Pages Required to Cache Index:           (   3   3   12 )
Processing Used to Search Index:         (   7   7   9 )

Key 0 Bucket Size                       (1-63) [3]      :

```

27. Then give the key a name (**WORLD_DESTINATION**).

```

Key 0 Name                               (1-32 chars) [null]
: WORLD_DESTINATION

```

28. You are then asked whether you want global buffers. Accept the default value (**NO**).

```

Global Buffers desired                   (Yes/No) [No]      :

```

29. All the required questions have been answered to define the first key. You will see a message like this. Press the Enter key.

```

The Depth of Key 0 is Estimated to be No Greater
than 1 Index levels, which is 2 Total levels.

```

```

Press RETURN to continue (^Z for Main Menu)

```

30. You can now define the second key (key 1). Notice that the options available are different from those for key 0. Accept the default value (**Line**).

Key 1 Graph Type Selection

```

Line   Bucket Size vs Index Depth      as a 2 dimensional plot
Fill   Bucket Size vs   Load Fill Percent   vs Index Depth
Key    Bucket Size vs           Key Length      vs Index Depth

Graph type to display                   (Keyword) [Line] :

```

31. Enter the following information for key 1, which is defined on **DESTINATION_CODE** only. (Details are given in steps 14 to 19). Accept the default values, except for

```

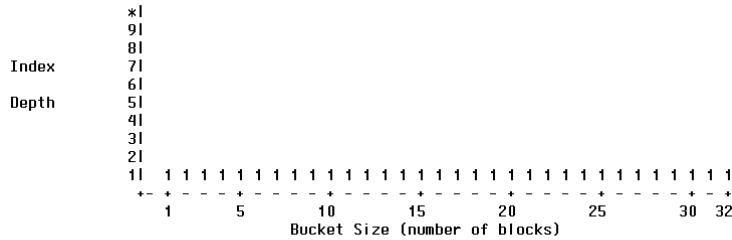
Key 1 Length: 10
Key 1 Position: 3

```

```

(Bin2 Bin4 Bin8 Int2 Int4 Int8 Decimal String Collated
 Dbin2 Dbin4 Dbin8 Dint2 Dint4 Dint8 Ddecimal Dstring Dcollated)
Key 1 Data Type (Keyword)[Str] :
Key 1 Segmentation desired (Yes/No)[No] :
Key 1 Length (1-43)[FD] : 10
Key 1 Position (0-33)[13] : 3
Key 1 Duplicates allowed (Yes/No)[Yes] :
Data Key Compression desired (Yes/No)[Yes] :
Index Compression desired (Yes/No)[No] : █
    
```

32. You will see a graph and information about key 1. Enter the value **FD** to continue.



```

PV-Prolog Version      3 KT-Key 1 Type      String EM-Emphasis Flatter ( 3)
DK-Dup Key 1 Values   Yes KL-Key 1 Length   10 KP-Key 1 Position   3
RC-Data Record Comp  0% KC-Data Key Comp  0% IC-Index Record Comp 0%
BF-Bucket Fill       100% RF-Record Format   Fixed RS-Record Size   43
LM-Load Method      RMS_Puts IL-Initial Load  0 AR-Added Records    200
(Type "FD" to Finish Design)
Which File Parameter (Mnemonic)[refresh] : █
    
```

33. You will then see some additional information, after which you should accept the default bucket size

```

Emphasis Used In Defining Default: ( Flatter_files )
Suggested Bucket Sizes: ( 3 3 12 )
Number of Levels in Index: ( 1 1 1 )
Number of Buckets in Index: ( 1 1 1 )
Pages Required to Cache Index: ( 3 3 12 )
Processing Used to Search Index: ( 8 8 10 )

Key 1 Bucket Size (1-63)[3] :
    
```

34. Do not allow changes. This means that key data cannot be changed once it has been added to the data file:

```

Key 1 Changes allowed (Yes/No)[No] :
    
```

35. Call the key **DESTINATION**

```

Key 1 Name (1-32 chars)[null]
: DESTINATION
    
```

36. Press the Enter key to continue.

The Depth of Key 1 is Estimated to be No Greater than 1 Index levels, which is 2 Total levels.

Press RETURN to continue (^Z for Main Menu)

37. The FDL editor main menu appears. View the FDL you have created from within the FDL editor (VIEW on the main menu). Check that it contains the information shown at the start of this section.

38. Save the FDL and exit the FDL editor by typing **exit** (the Quit command will not save the file).

39. View the entire FDL file (example follows).

\$ TYPE WORLD_DESTINATION.FDL

```
TITLE      "WORLD_DESTINATION"
IDENT      "29-JUL-1997 11:02:40  OpenVMS FDL Editor"

SYSTEM
SOURCE          "OpenVMS"

FILE
NAME            "<device><directory>WORLD_DESTINATION.DAT"
ORGANIZATION    indexed

RECORD
CARRIAGE_CONTROL carriage_return
FORMAT          fixed
SIZE            43

AREA 0
ALLOCATION       42
BEST_TRY_CONTIGUOUS yes
BUCKET_SIZE     3
EXTENSION       9

AREA 1
ALLOCATION       6
BEST_TRY_CONTIGUOUS yes
BUCKET_SIZE     3
EXTENSION       3

AREA 2
ALLOCATION       24
BEST_TRY_CONTIGUOUS yes
BUCKET_SIZE     3
EXTENSION       9
```

```

KEY 0
  CHANGES                no
  DATA_AREA              0
  DATA_FILL              100
  DATA_KEY_COMPRESSION  yes
  DATA_RECORD_COMPRESSION yes
  DUPLICATES              no
  INDEX_AREA              1
  INDEX_COMPRESSION      no
  INDEX_FILL              100
  LEVEL1_INDEX_AREA      1
  NAME                    "WORLD_DESTINATION"
  PROLOG                  3
  SEG0_LENGTH             13
  SEG0_POSITION           0
  TYPE                    string

KEY 1
  CHANGES                no
  DATA_AREA              2
  DATA_FILL              100
  DATA_KEY_COMPRESSION  yes
  DUPLICATES              yes
  INDEX_AREA              2
  INDEX_COMPRESSION      no
  INDEX_FILL              100
  LEVEL1_INDEX_AREA      2
  NAME                    "DESTINATION"
  SEG0_LENGTH             10
  SEG0_POSITION           3
  TYPE                    string

$

```

Tutorial: Create a New RMS Data File

Using the FDL file that you have just defined, you can now create an empty data file.

1. At the DCL command prompt type:
\$ CREATE/FDL=WORLD_DESTINATION.FDL

This will create a data file using the definition contained in the **WORLD_DESTINATION.FDL** file definition. The file specification of the data file that is created is contained in the FDL. (You can overwrite this by typing the name of a new data file in the command line).

2. Verify that the file has been created. At the DCL command prompt type:
\$ DIRECTORY /FULL EASYSOFT_SQL_DATA:WORLD_DESTINATION.DAT

You will see all the details of the file that has been created.

Add a Further Key to an Indexed File

WARNING

NEVER modify the key of any file which is used by applications other than Easysoft unless you are absolutely sure that none of those applications will be affected by the change you make.

If you wanted to add another key to a data file, this is what you would do.

1. \$ **EDIT/FDL** <FDL file to modify>
2. At the top level menu, select the Add option. The Legal Primary Attributes menu appears.

Legal Primary Attributes

```
ACCESS  attributes set the run-time access mode of the file
AREA x  attributes define the characteristics of file area x
CONNECT attributes set various RMS run-time options
DATE    attributes set the date parameters of the file
FILE    attributes affect the entire RMS data file
KEY y   attributes define the characteristics of key y
NETWORK attributes set-run time network access parameters
RECORD  attributes set the non-key aspects of each record
SHARING attributes set the run-time sharing mode of the file
SYSTEM  attributes document operating system-specific items
TITLE   is the header line for the FDL file
```

Enter Desired Primary (Keyword) [FILE] : KEY 2

3. Enter the value **key n**, where n represents the key number. The Legal KEY n Secondary Attributes menu appears.

Legal KEY 2 Secondary Attributes

CHANGES	yes/no	LEVEL1_INDEX_AREA	number
DATA_AREA	number	NAME	string
DATA_FILL	number	NULL_KEY	yes/no
DATA_KEY_COMPRESSION	yes/no	NULL_VALUE	char/num
DATA_RECORD_COMPRESSION	yes/no	POSITION	number
DUPLICATES	yes/no	PROLOG	number
INDEX_AREA	number	TYPE	keyword
INDEX_COMPRESSION	yes/no	SEgN_LENGTH	number
INDEX_FILL	number	SEgN_POSITION	number
LENGTH	number	COLLATING_SEQUENCE	string

Enter KEY 2 Attribute (Keyword) [-1] :

4. For each attribute you wish to add, enter that attribute and then complete the dialog that follows.
5. Select the Exit option to save the changes and then exit the FDL editor. (Select Quit to abandon changes).

6. Generate a data file. Select one of these two options:
- a) If the data file does not exist, or if it is empty, then generate a new data file (CREATE/FDL=<FDL to use>) or
 - b) If the data file exists, and if it contains records, use CONVERT/FDL to generate a new data file containing the new file organisation from an existing data file (see note below).

Note - Convert command

The convert command can be used to convert a data file which contains records into another one with a different file organisation (the data will be retained, but in a different format). As a precursor to this, there must be an FDL definition of the new file. This is what you would do to convert a file.

```
$ CONVERT/FDL=<FDL name>.FDL <input file> <output file>
```

This command creates a new file, according to the specification in the FDL file. The input data file and the output data file names can be the same, in which case only one data file remains after the format of the records has been changed. If the input and output file names are different, the old data file remains after the conversion.

Summary

In this module you have learnt how to

- ✦ determine RMS file and record structure using an FDL (generated using ANALYZE/RMS_FILE/FDL)
- ✦ determine the indexes available using an FDL
- ✦ create a new FDL file (EDIT/FDL)
- ✦ create a new empty data file using an FDL file (CREATE/FDL)
- ✦ add a key using an FDL and the Convert utility

Review Questions

1. What does FDL stand for?
2. What does an FDL file describe?
3. What DCL command is used to generate an FDL from an existing data file?
4. Although FDLs can be used to determine the record structure of a data file, there is a simpler method to obtain the information needed for the Easysoft File Definition dialog box. What is this simpler method?
5. What is the name of the section in an FDL that describes the indexes in an RMS file and how many of these sections are there?
6. What RMS utilities are there to determine the field structure of a record.
7. What DCL command do you use to create an empty data file from an FDL?

Exercises

All the answers can be obtained from the notes. It's the quickest way of answering the questions, but you probably won't learn much; perform the operations where appropriate. Answers to these exercises are on page 7-24.

File information

1. The CURRENCY_RATE file exists in EASYSOFT_SQL_DATA directory. Write down the following for the file:

File organisation

Record type

Record size

2. What's the easiest way of determining this information?

Keys

Use ANAL/RMS/FDL to generate an FDL to answer these questions.

1. How many keys are there in the CURRENCY_RATE file?
2. What are the key definitions (i.e. starting position, length)?
3. What fields do the keys correspond to? (Fields shown below).

CURRENCY_CODE 4 BYTES	CURRENCY_RATE 8 BYTES	CURRENCY_DATE 11 BYTES
--------------------------	--------------------------	---------------------------

Modify the Key

From a previous exercise you have determined that a segmented key is appropriate on CURRENCY_CODE and CURRENCY_DATE in the CURRENCY_RATE file. Follow these step-by-step instructions to change the key in the file.

WARNING

NEVER modify the key of any file which is used by applications other than Easysoft unless you are absolutely sure that none of those applications will be affected by the change you make.

SEGMENT 0 4 BYTES		SEGMENT 1 11 BYTES
CURRENCY_CODE 4 BYTES	CURRENCY_RATE 8 BYTES	CURRENCY_DATE 11 BYTES

1. Use EDIT/FDL to modify the CURRENCY_RATE.FDL file
2. Select the **MODIFY** option
3. Enter the value **KEY 0**
4. Modify the following attribute
DUPLICATES **no**
5. Modify the following attribute
NAME **CODE_AND_DATE**
6. Return to the main menu
7. Add the following attribute (ADD, KEY 0)
SEG1_LENGTH **11**
8. Return to the main menu
9. Add the following attribute (ADD, KEY 0)
SEG1_POSITION **12**
10. Exit the FDL editor (do not use Quit - changes will be lost).
11. Use CONVERT/FDL to convert the data file structure. (The files are in EASYSOFT_SQL_DATA - this should prefix the file name).

Review Answers

1. What does the acronym FDL stand for?

File Definition Language

2. What does an FDL file describe?

It describes the file structure of a data file, including the indexes available.

3. What DCL command is used to generate an FDL from an existing data file?

ANAL/MS/FDL <data file specification>

4. Although FDLs can be used to determine the record structure of a data file, there is a simpler method to obtain the information needed for the Easysoft File Definition dialog box. What is this simpler method?

Use DIR/FULL.

5. What is the name of the section in an FDL that describes the indexes in an RMS file and how many of these sections are there?

KEY. There is one KEY section for each index.

6. What RMS utilities are there to determine the field structure of a record.

None. Fields in a record have no meaning in the RMS structure.

7. What DCL command do you use to create an empty data file from an FDL?

CREATE/FDL=<FDL file name>

Answers to Exercises

File information

1. Write down the following for the CURRENCY_RATE file:

File organisation indexed

Record type fixed

Record size 23

2. What's the easiest way of determining this information?

Use DIR/FULL

Keys

1. How many keys are there in the CURRENCY_RATE file?

One

2. What are the key definitions?

Starting position: 0

Length: 4

3. What fields do the keys correspond to?

CURRENCY_CODE

Supplement: FDL File Structure

An FDL file is a text file which consists of a series of file sections, each section describing some aspect of a data file (which possibly does not yet exist).

Sections (*primary attributes*) may be decomposed using *secondary attributes* and some of these secondary attributes have a further level of attributes which are called *qualifiers*. Secondary attributes can be either create-time attributes or run-time attributes.

When you look at an FDL file for the purposes of determining a file organisation, most of these sections (if they exist) are irrelevant. However, since they may appear in the FDL, they are all briefly described here.

The file sections must appear in the following order; some sections are optional.

TITLE

If you use EDIT/FDL to create the FDL file, then you are asked for a title that will be placed at the start of the FDL file. It is used for comment purposes only.

IDENT

If either EDIT/FDL or ANALYZE/RMS_FILE are used to create an FDL file, they place the IDENT attribute in the file. This contains the date and time of creation and the name of the utility that created the file.

SYSTEM

This section consists of system identification information and can be used to help document the FDL file. All the secondary attributes are run-time attributes.

FILE

This section allows you to specify file processing and file-related aspects for the data file.

The attributes BEST_TRY_CONTIGUOUS, BUCKET_SIZE, CONTIGUOUS and EXTENSION have corresponding attributes in the AREA section which over-ride any values specified in the FILE section.

DATE

This section is used to specify dates and times of various file characteristics. In general, you should let the system specify these values.

RECORD

This section contains secondary attributes that define various controls for the record. All the secondary attributes are create-time attributes.

ACCESS

This allows you to specify various file processing operations by assigning values to the secondary attributes. All the secondary attributes are run-time attributes.

NETWORK

This section sets network access parameters. All the secondary attributes are run-time attributes.

SHARING

This section allows you to specify attributes which control the options for sharing the file. All the secondary attributes are run-time attributes.

CONNECT

This section specifies application-dependent run-time attributes that are related to access and performance.

AREA

This section is an RMS-specific region of an indexed file that cannot be manipulated using a high-level programming language. If you want to create or manipulate areas in a file, you must include the AREA primary attribute in an FDL file. All AREA secondary attributes are create-time attributes.

Most Area secondary attributes have corresponding FILE secondary attributes; any values specified here over-ride those specified in the FILE section.

KEY

For each key of an indexed file, there must be a KEY section. This section must be numbered. The value for the primary key must be 0 (KEY 0). Secondary keys can have values between 1 and 254. All KEY secondary attributes are create-time attributes; they are described in the next section.

ANALYSIS_OF_AREA

This section only appears in FDL files that are created using the Analyze/RMS_FILE utility (ANALYZE/RMS_FILE command). There is only one secondary run-time attribute, RECLAIMED_SPACE, which is automatically generated.

ANALYSIS_OF_KEY

This section only appears in FDL files that are created using the Analyze/RMS_FILE utility (ANALYZE/RMS_FILE command). The Edit/FDL utility uses values in this section in its Optimize script. All the secondary attributes are run-time.

Description of KEY Secondary Attributes

CHANGES

This switch allows an update operation to change the value of the key. This is not allowed for a primary key, so this value must always be NO. For alternate keys the default is NO, but it can be changed.

COLLATING_SEQUENCE

This defines the name of the collating sequence that defines the sort order for the key.

DATA_AREA

Defines the area in which you the data records in an indexed file which has multiple areas.

The value is an integer in the range 0 to 254, and it must be the same number as that assigned to the area in an AREA section.

DATA_FILL

This attribute establishes the percentage of bytes in each bucket in the area that is to be populated initially.

DATA_KEY_COMPRESSION

This switch controls whether repeating leading and trailing character are compressed in the primary key. The default is YES; for compression to occur, however, the file must be defined as a Prolog 3 file (see entry below).

DATA_RECORD_COMPRESSION

This switch controls whether repeating character are compressed in the data. The default is YES; for compression to occur, however, the file must be defined as a Prolog 3 file (see entry below).

DUPLICATES

This switch controls whether duplicate keys are allowed in the indexed files. When duplicate keys are allowed, an attempt to write a record where the key would be a duplicate results in an error.

INDEX_AREA

This defines the area where the index levels (other than 1) are placed in an indexed file with multiple areas.

INDEX_COMPRESSION

This controls whether leading repeating characters in the index are compressed.

INDEX_FILL

This sets the number of bytes in each index bucket.

LENGTH

This sets the length of the key in bytes.

LEVEL1_INDEX_AREA

In an indexed file with multiple areas, this defines where the level 1 index is placed.

NAME

This string attribute can be used to assign a name to a key.

NULL_KEY

This controls whether null key values are allowed in an alternate string keyfield.

NULL_VALUE

This attribute specifies the null value. The default is the ASCII null character.

POSITION

This defines the byte position of the start of the key field. Primary keys work best if they start at byte position 0.

PROLOG

This defines which one of the three Prolog types will be used.

SEG_n_LENGTH

This defines the length of the key segment in bytes. and is used with the SEG_n_POSITION attribute when the key is segmented. n may be in the range 0 to 7, and the first segment in the key must be numbered 0. Segmented keys must be of STRING type.

SEG₀_POSITION

This indicates the position of the key within the segment. The corresponding name in the Field Details dialog box is "Offset"

TYPE

This indicates the key type in the index and must have one of the following values (which are sorted into ascending and descending groups).

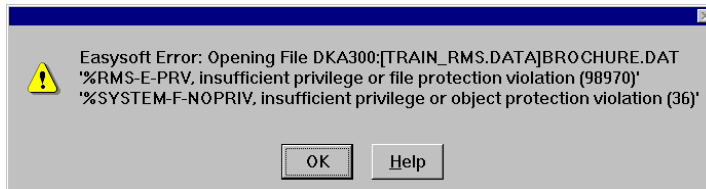
Ascending: Bin2, Bin4, Bin8, Collated, Decimal, Int2, Int4, Int8, String

Descending: Dbin2, Dbin4, Dbin8, Dcollated, Ddecimal, Dint2, Dint4, Dint8, Dstring

Supplement: File Protection

In the module entitled “Introduction to RMS Files” you saw how a file had read, write, execute and delete privileges for each of four categories of user, namely, System, Owner, Group and World.

If a user does not have the correct RMS privileges to access a file, then when an attempt is made to read or write to that file using Easysoft ODBC, a message similar to this will appear:



To resolve the problem, you could

- change the protection on the file
- allow the user access by means of an Access Control List (ACL)

Change File Protection

To view the file protection that is assigned to a file, use the SHOW SECURITY command:

```
$ SHOW SECURITY EASYSOFT_SQL_DATA:BROCHURE.DAT

DKA0: [TRAINER.RMS.DAT.TRAINER]BROCHURE.DAT;2 object of class FILE
Owner: [TRAINER]
Protection: (System: RWED, Owner: RWED, Group, World)
Access Control List: <empty>
```

In this case, you can see that the Group and World users do not have any privileges to the file. To allow Group users read and write access and World users read access, you would do the following:

```
$ SET SECURITY /PROTECTION=(G:RW,W:R) EASYSOFT_SQL_DATA:BROCHURE.DAT
```

If you now view the security, you will see

```
Protection: (System: RWED, Owner: RWED, Group: RW, World: W)
```

Use Access Control List

An access control list is a collection of entries that define the access rights a user or group of users has to an object, such as a file or directory. ACLs can be created by default when the object is created, they can be created by the security administrator, or they can be created by the user who has control access.

The use of ACLs is optional. They can be created and displayed using the SET SECURITY and SHOW SECURITY commands (you have seen examples of these commands in the previous section). There is also an ACL editor.

Say that in the general case for a particular file, Group users have read and write privileges, and World users do not have any access to the file, thus:

```
$ SHOW SECURITY TESTFILE.DAT
```

```
DKA300:[MIKEU]TESTFILE.DAT;1 object of class FILE
  Owner: [MIKEU]
  Protection: (System: RWED, Owner: RWED, Group: RE, World)
  Access Control List: <empty>
```

In the example that follows, we will grant read and write access to a World user called Richard, and we will prevent one Group user, Carolyn, from accessing the file in any way. To do this, we will create an ACL with the appropriate entries using the SET SECURITY command.

```
$ SET SECURITY /ACL=((IDENTIFIER=RICHARD, ACCESS=READ+WRITE), -
_$(IDENTIFIER=CAROLYN, ACCESS=NONE)) TESTFILE.DAT
```

Each entry in the ACL is of the form

(IDENTIFIER=<name>, ACCESS=<privilege [+privilege]...>)

```
$ SHOW SECURITY TESTFILE.DAT
```

```
DKA300:[MIKEU]TESTFILE.DAT;1 object of class FILE
  Owner: [MIKEU]
  Protection: (System: RWED, Owner: RWED, Group: RE, World)
  Access Control List:
    (IDENTIFIER=RICHARD,ACCESS=READ+WRITE)
    (IDENTIFIER=CAROLYN,ACCESS=NONE)
```

```
$
```


8. Easysoft Administration on the PC

This module deals with the administration of the Easysoft system on the PC. First you will do a paper exercise in which you determine what exactly you will enter into the PC Administrator. Then you will use the Administrator to input this.

In this module you will learn how to

- ✦ Use the Easysoft PC Administrator to define RMS files so they can be used by ODBC-compliant applications
- ✦ Import and export file definitions from the Easysoft PC Administrator

Contents

Determine Field Lengths and Offsets	8-2
Stages of Defining a Server File	8-3
Starting the Administrator	8-5
Subsequent Administrator Logon	8-6
File Definition Tutorial	8-7
Step 1. Download Easysoft Catalog	8-8
Step 2. File Definitions	8-9
Step 3. Field Definitions	8-10
Step 4. Database Definitions	8-13
Step 5. Table Definitions	8-14
Step 6. Column Definitions	8-16
Step 7. User Definitions	8-17
Step 8. Create Database Privileges	8-18
Step 9. Upload Easysoft Catalog	8-19
Populate Server File	8-21
Importing and Exporting Definitions	8-23
Exporting Definitions	8-23
Importing Definitions	8-24
Summary	8-27
Review Questions	8-28
Review Answers	8-30
Supplement: Easysoft PC Administrator Installation	8-31
Download the Software	8-31
Administrator Installation Steps	8-31

Determine Field Lengths and Offsets

When you use the Easysoft PC Administrator to define an RMS file in SQL terms, you need to give it information (amongst other things) about the RMS file and the fields in that file. You have seen the structure of the records in the CURRENCY_RATE.DAT data file in a previous module. It has been defined as:

CURRENCY_CODE 4 BYTES	CURRENCY_RATE 8 BYTES	CURRENCY_DATE 11 BYTES
--------------------------	--------------------------	---------------------------

The file is indexed, with fixed length records. All the fields in a record are string fields.

You may recall that in a previous module, the starting position (offset) of each of the fields was listed. For clarity, this information is repeated here.

<u>Field name</u>	<u>Field length</u>	<u>Offset</u>
CURRENCY_CODE	4	0
CURRENCY_RATE	8	4
CURRENCY_DATE	11	12

Record size = 23 bytes

When you come to define the file using the Easysoft PC Administrator, the offsets will be calculated automatically, but it's useful to know beforehand what they should be, so you can check for typing errors.

There is another indexed file, CURRENCY_MASTER.DAT, containing the following fields all of which are string fields:

CURRENCY_CODE 4 BYTES	TITLE 30 BYTES	UNITS_PLURAL 12 BYTES	UNITS_SINGULAR 12 BYTES
--------------------------	-------------------	--------------------------	----------------------------

Exercise - field lengths and offsets

Write down the field length and the starting position (offset) of each of the fields in the CURRENCY_MASTER.DAT file:

<u>Field name</u>	<u>Field length</u>	<u>Offset</u>
CURRENCY_CODE		
TITLE		
UNITS_PLURAL		
UNIT_SINGULAR		

What is the record size?

Answers on page 8-30.

Now you can define the RMS file using the Easysoft PC Administrator. The steps are outlined below.

Stages of Defining a Server File

This section lists the basic functions to perform within the Easysoft Administrator to define a Server file for use through the Easysoft ODBC Driver (these steps are described in detail in the section called "File Definition Tutorial" on page 8-7). Before you can define a file on the Server a catalog should be created on the Server and a data source associated with that catalog should be defined using the Easysoft ODBC Setup dialog box.

- | | | |
|---|---------------------------|--|
| 1 | Download Easysoft Catalog | Ensures that the current local copy of the Easysoft catalog is as up-to-date as possible |
| 2 | File Definition | Describes the structure of a file on the Server |
| 3 | Field Definitions | Describes the fields within a file on the Server |
| 4 | Database Definition | If a database does not exist, then create one |
| 5 | Table Definition | SQL table name to map on to a File Definition. Assign table security if required |
| 6 | Column Definition | SQL columns to map on to fields of a file on the Server |

- | | | |
|---|----------------------------|--|
| 7 | User Definition | Create users and assign passwords |
| 8 | Create Database Privileges | Assign user security for a Database |
| 9 | Upload Easysoft catalog | Update the Easysoft catalog on the Server with the newly entered PC data |

Note: the terms “Easysoft catalog” and “System catalog” are synonymous.

Once these steps have been performed, the data can be accessed using an ODBC-compliant application.

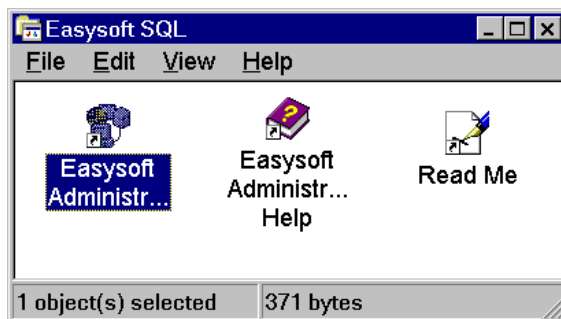
The following sections take you through starting the Administrator and the steps of defining a file on a server.

Starting the Administrator

The Easysoft PC Administrator requires a password for access; this section describes how to set this up.

1. To start the Easysoft Administrator select the Easysoft Administrator icon.

This can be found in the **Programs** menu under the **Start** button or in the Easysoft SQL Program Group.

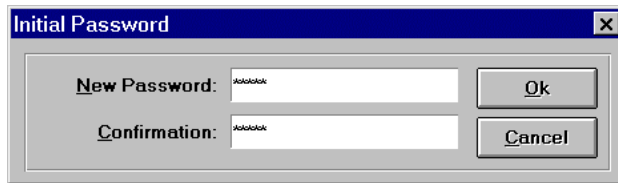


2. After starting the Easysoft Administrator the greetings screen is displayed for a few seconds.



This screen displays version and licence information
<Administrator User> and <Company Name> show the names that were entered during the setup process.

3. The *first* time the Administrator is used after it has been installed, the user is presented with the Initial Password dialog box. It is used to define a password which controls access to the Easysoft Administrator.



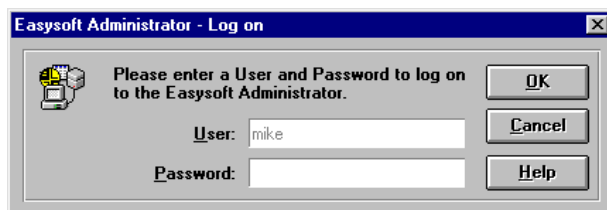
4. The entry fields are initially blank. The user name (not shown in this dialog box) associated with the password is the name specified in the User Details dialog box during the installation procedure. Insert the new password and confirmation and then select the **OK** button. Passwords are not case sensitive.

Write the new password here: _____

5. Once the new password and confirmation have been entered correctly the user receives a message: Password changed successfully.
6. After clicking **OK** you are logged in and presented with the Easysoft Administrator. Since a local copy of the Easysoft catalog which resides on the Server is needed to use the Administrator, the Download System Catalog dialog box is immediately presented. This happens only the first time you log on (except if you do not download the Easysoft catalog at this stage, then the next time you log on, the procedure is repeated). See "File Definition Tutorial" on page 8-7 for details.

Subsequent Administrator Logon

1. All but the first time that the Easysoft Administrator is used the Logon dialog box is automatically presented.



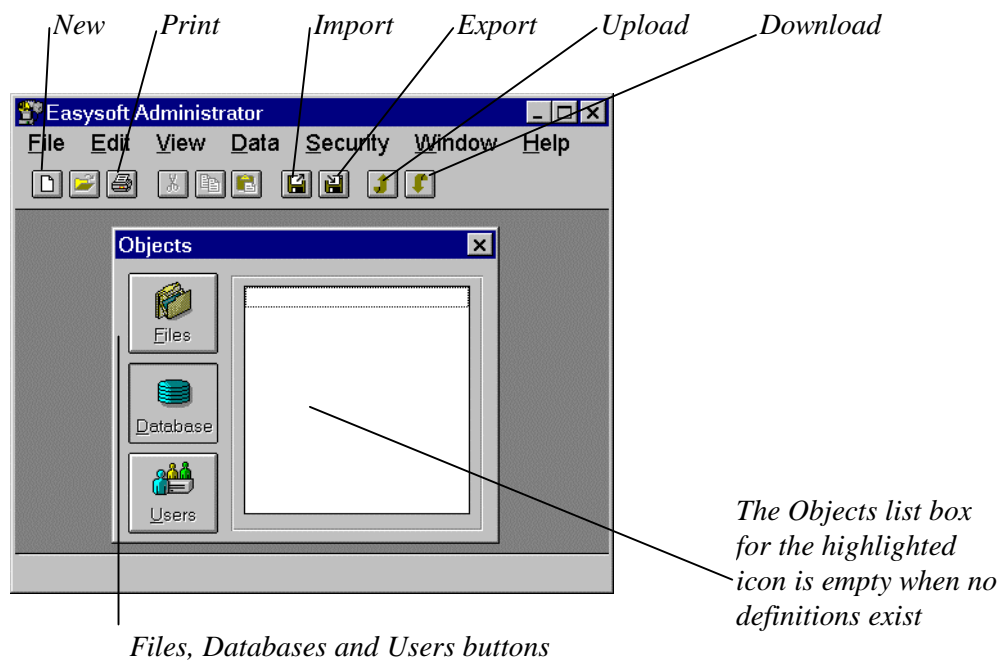
2. Insert the Administrator password that was created using the Initial Password dialog box and then select the **OK** button. (The User entry field is always greyed out, as it cannot be changed; it shows the same name that was entered in the User Details dialog box. If the password is incorrect, a message is generated. After three incorrect attempts the program closes).

File Definition Tutorial

This section gives step-by-step instructions on how to use the Easysoft Administrator to define a data file which resides on the Server. Once the tutorial is completed you will be able to use the defined file. You can stop part of the way through, and resume later if you wish. You do not need to save changes to the catalog on the Server; changes are saved locally, and when the tutorial is complete, you can upload the new catalog contents.


This is the Easysoft Administrator and the Objects screen which is the central control screen of the Administrator.

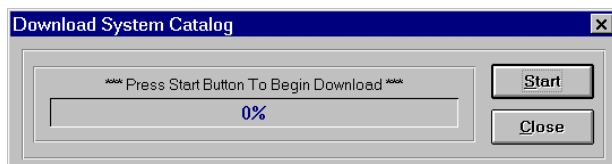
Easysoft Administrator with Objects dialog box (no definitions)



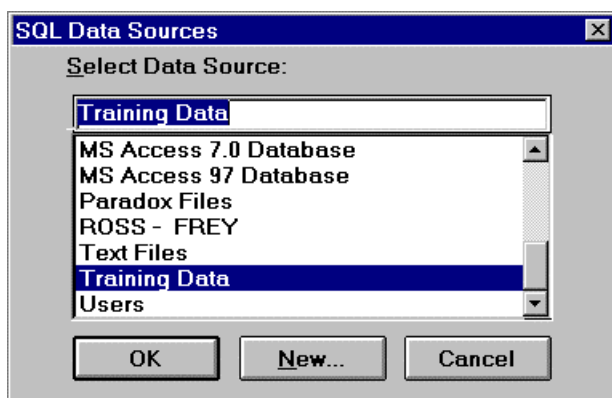
The stages of defining a file have already been outlined. The step numbers in this section correspond to those on page 8-3.

Step 1. Download Easysoft Catalog

1. The first action to perform before any definitions can be entered locally on the PC is to download the relevant Easysoft catalog from the Server. Thus, the *first time* that the Administrator is used, the Download System Catalog dialog box is automatically presented. (In general, it is accessed either by selecting the **Download** option from the **Data** pulldown menu or by selecting the **Download** button () on the Administrator toolbar).



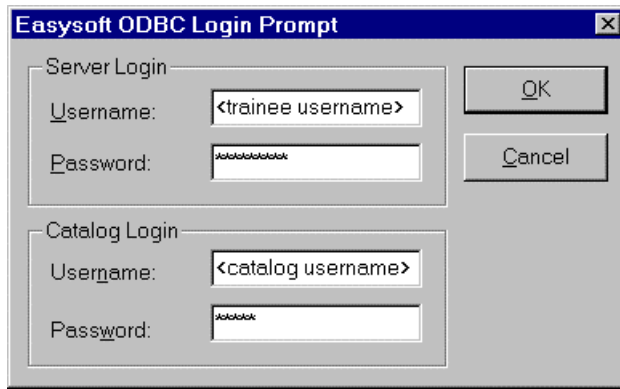
2. Select the **Start** button to begin the download. You are presented with a warning stating that all local definitions will be deleted; click **Yes** to continue. Then the SQL Data Sources dialog box appears.



3. Highlight the TRAINING DATA data source and click **OK**.

Note: in the download process you are not asked for an Easysoft catalog name - rather, a list of defined data sources is presented and you select one which uses the required catalog. A wise precaution is to be fully aware of the relationship between data sources and catalogs which exist.

4. The Easysoft ODBC Login Prompt *may* appear. If it does, ensure that the Username and Password details are complete, and then click the **OK** button. (The appearance or otherwise of the Login Prompt depends upon the value of the “Prompt for Login” option on the Easysoft ODBC Settings dialog box).



5. The Download System Catalog dialog box re-appears, but this time the buttons are greyed out, and as the catalog tables are downloaded the percentage bar moves to the right. Once the percentage bar has reached 100% the download operation has completed successfully. Press the **Close** button.


Step 2. File Definitions

A file definition is essentially a description of the record structure of a physical file. Here you will define the structure of a data file that was created in a previous module.

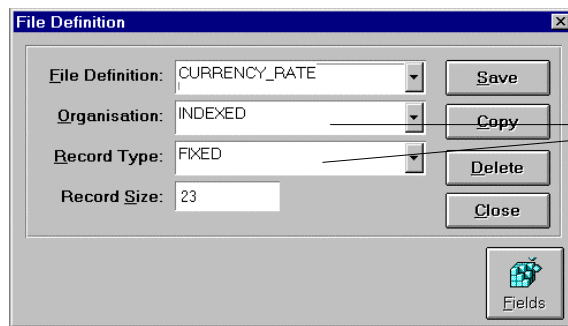
1. Select the **Files** button from the Objects screen.



A list of current file definitions is displayed.

2. Insert a new file definition by either selecting **New** from the **File** pulldown menu or clicking the **New** button () on the toolbar. The File Definition dialog box appears - initially the first three entry fields are empty, and the Record Size field defaults to zero.

Complete the dialog box with the relevant information you determined in the paper exercise.



The 'File Definition' dialog box contains the following fields and buttons:

- File Definition:** CURRENCY_RATE (dropdown menu)
- Organisation:** INDEXED (dropdown menu)
- Record Type:** FIXED (dropdown menu)
- Record Size:** 23 (text input)
- Buttons:** Save, Copy, Delete, Close
- Fields:** A button with a grid icon and the label 'Fields' at the bottom right.

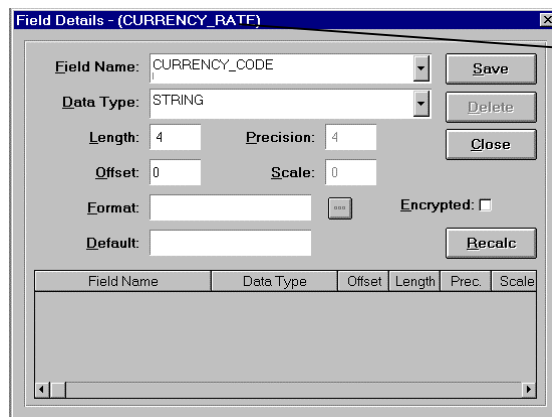
Select from the pulldown menu or type first letter

- Once your definition matches the dialog box above select the **Save** button. The appearance of the **Fields** button does not change when it is enabled. However, if it is disabled and if you click on it, a message appears stating that it cannot be used.

Step 3. Field Definitions

The Field Details dialog box allows you to define the fields contained within a file definition. In this exercise, you will define the fields for CURRENCY_RATE.

- Select **Fields** from the File Definition dialog box. The Field Details dialog box appears. Complete it as follows:



The 'Field Details - (CURRENCY_RATE)' dialog box contains the following fields and buttons:

- Field Name:** CURRENCY_CODE (dropdown menu)
- Data Type:** STRING (dropdown menu)
- Length:** 4 (text input, greyed out)
- Precision:** 4 (text input, greyed out)
- Offset:** 0 (text input)
- Scale:** 0 (text input)
- Format:** (text input)
- Encrypted:**
- Default:** (text input)
- Buttons:** Save, Delete, Close, Recalc

Field Name	Data Type	Offset	Length	Prec.	Scale

Details on screen refer to this File Definition.

- Length and Precision are defaults which depend on the data type; if they cannot be changed they are greyed out (as in this case).
- Offset is the starting position of the field with respect to the start of the file. It is automatically calculated using the lengths of the previous fields.
- Encrypted and the use of **Recalc** are not used in this tutorial.

- d) Once all the field details are correct press the **Save** button. A row appears towards the bottom of the dialog box showing the details of this field.
2. Enter the following information for the CURRENCY_RATE field and then click the **Save** button.

Field Name	Data Type	Offset	Length	Prec.	Scale
CURRENCY_CODE	STRING	0	4	4	0

3. Enter the following information for the CURRENCY_DATE field and then click the **Save** button.

Field Name	Data Type	Offset	Length	Prec.	Scale
CURRENCY_CODE	STRING	0	4	4	0
CURRENCY_RATE	DOUBLE	4	8	15	0
CURRENCY_DATE	DATE	12	11	11	0

Defined Fields area shows details of all defined fields.

Use the scroll bar to see Format, Default etc.


Note: Default is used to specify the default value that will be entered into a new record if a user does not specify a date. There is no validation of the entry to the field here; it is possible to enter a date which does not match the specified format.

4. Once all the field details have been entered, **Save** the details and then press the **Close** button. Control passes back to the File Definition dialog box.

If you notice a mistake in any of the details of a previously entered field, you can correct the information by either selecting the field from the list of fields in the Field Name drop-down combo box or by typing the name directly into the text box or by highlighting the field in the Defined Fields area. Details for the field appear; correct these and press **Save**.

Note: defining dates

A common misconception is that you can use the Administrator to change the appearance of dates and times when viewed through ODBC-applications. This is not the case.

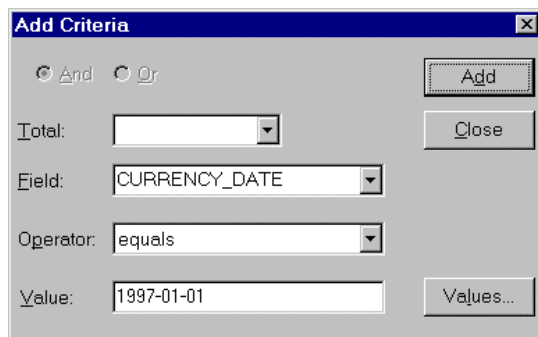
The purpose of the Format button () is to define the existing structure of RMS data, so that when an ODBC application requires access to the data, the Easysoft software knows the correspondence between the ODBC format and the RMS format. A popular format for dates in RMS is DD-MMM-YYYY, and this is the default option that is presented, so you do not need to use the Format button.

To change the underlying structure of RMS data you must use an RMS utility.

An additional point of confusion is that an ODBC application may not necessarily show dates and times in the standard format that is used in ODBC function calls. For example, in Microsoft Excel the Format Cells dialog box can be used to change the format of dates. The example below shows how the date has been formatted to appear as Month Day, Year. Note the different format in the formula bar.

B2		01-01-1997		
	A	B	C	
1	CURRENCY_CODE	CURRENCY_DATE	CURRENCY_RATE	
2	AED	January 1, 1997	162	
3	ALL	January 1, 1997	203.172	

If you are familiar with the ODBC format of SQL dates, you will notice that the date in formula bar does not correspond to the ODBC format. However, the underlying query does indeed contain the correct ODBC format (YYYY-MM-DD) as is shown in the Add Criteria dialog box (in Microsoft Query).



Add Criteria

And Or Add

Total: ▼ Close

Field: CURRENCY_DATE ▼

Operator: equals ▼

Value: 1997-01-01 Values...

Step 4. Database Definitions

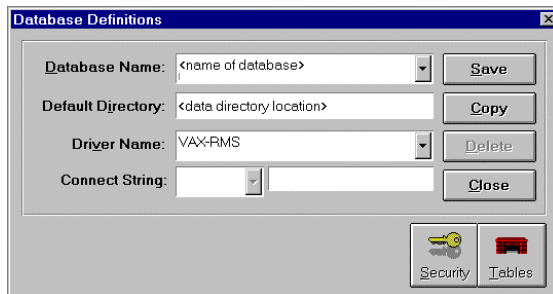
A database is a collection of tables. A database must be defined prior to defining tables. In this case, the TRAINING database already exists, so there is nothing to do. However, for completeness, we show you how to define a database.

1. From the Object screen select the **Database** button.



The current list of database definitions is displayed.

2. Insert a new definition by either selecting **New** from the **File** pulldown menu or by clicking the **New** button on the toolbar. You are presented with the Database Definitions dialog box. The entry fields are initially empty. Complete the dialog box as follows:



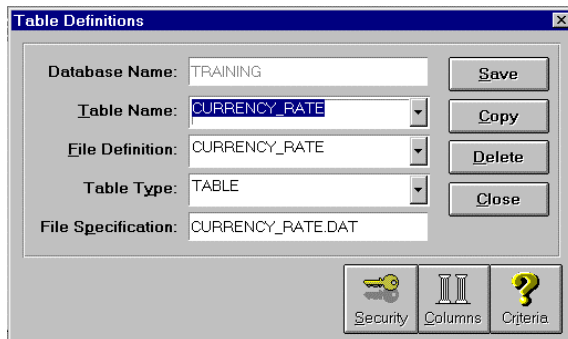
Note: Connect String is not used with Easysoft ODBC for RMS.

3. Press the **Save** button.

Step 5. Table Definitions

1. Double-click on the TRAINING database name to open the Database Definitions dialog box.
2. Press the **Tables** button. The Table Definition dialog box appears (it will not be empty, but will show existing table definitions). This is used to map an SQL table onto the previously defined file definition (available in the File Definition pulldown, and created "Step 2. File Definitions", page 8-9).

Select the CURRENCY_RATE file definition and add a table name to map the file to a table. (Include the file specification with the path, unless a default directory has been defined - as it has in this case). Complete the dialog box as follows:

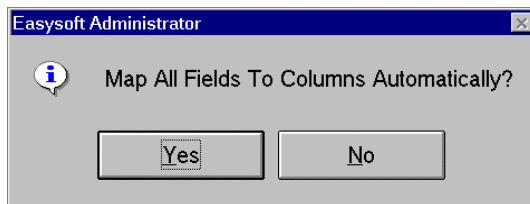


Database Name is always greyed out, as it cannot be changed using this dialog box.

Table Security is not used in this tutorial - see "Note - table security" on page 8-15 for an explanation of how to use it.

Table Criteria is not used in this tutorial - see "Note - table criteria" on page 8-16 for an explanation of how to use it.

3. Press the **Save** button. Because this is a new table, you are presented with the mapping options dialog box.



4. Select **No**; you will have to perform Step 6 manually (when you define the CURRENCY_MASTER file, you can select **Yes** at this stage).

Note - map fields automatically

Selecting **Yes** will automatically create columns within a table for all fields within a file definition. If **Yes** is selected then the manual operations involved in Step 6 are not required.

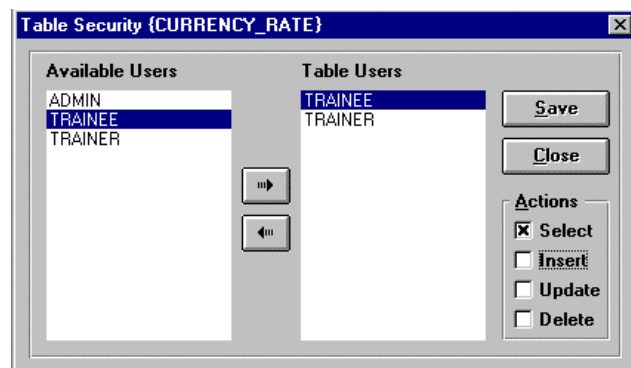
It matters not whether **Yes** or **No** is pressed; it's all a question of convenience. Will the task of mapping fields to columns be performed more efficiently by mapping all the fields automatically and then deleting those that are not needed or would it be better to manually map those fields that are required?



Whether you press **Yes** or **No**, control passes back to the Table Definitions dialog box.

Note - table security

The Table Security dialog box allows security privileges at table level to be applied to a particular user. It is accessed by selecting the **Security** button on the Table Definitions dialog box.

Table security is completely ignored if the **Security/Table** checkbox is not selected on the Database Security dialog box for the specified user (see Step 8).



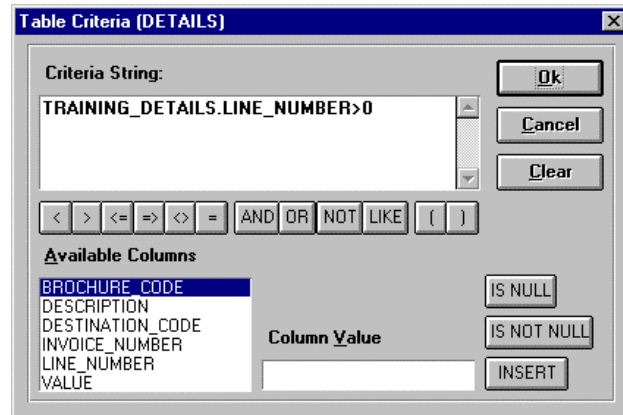
Giving privileges to a new user for an existing table is done by selecting the relevant user in the left-hand list box and selecting the  button. Deleting a user is the reverse process; select the user in the Table Users list and press the  button.

Once a user has been inserted into the list of table users, specific restrictions may be attached to that user. Select the table user and enter the actions allowed.

Select	Allows the user to read records from the specified table
Insert	Allows the user to insert records into the specified table
Update	Allows the user to update existing records in a table
Delete	Allows the user to delete records from a table

Note - table criteria

The Table Criteria dialog box is used to set up default criteria to eliminate records from an SQL table.

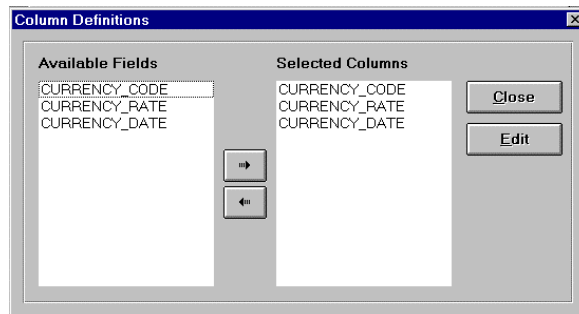


Criteria String shows the criteria. Any legal SQL is allowed, provided the length does not exceed 254 characters.

The criteria string can be entered directly, or built up using the options available from the various buttons.

Step 6. Column Definitions

1. Select the **Columns** button from the Table Definitions dialog box. The Column Definitions dialog box appears (here shown completed).



2. Highlight all of the fields and press the **Add** button.
3. Click **Close** to return to the Table Definitions dialog box. The definition of the data files is now complete, all that remains to do is to create and assign privileges to a user in order to access the new database.

4. Save the data.
5. Close the Table Definitions dialog box.
6. Close Database Definitions; the Object dialog box appears.

Exercises - CURRENCY_MASTER

There is an indexed data file, CURRENCY_MASTER.DAT, which you should define using the Easysoft PC Administrator. In the paper exercise on page 8-3 you determined the information you needed to complete this exercise. The information is repeated here for convenience.

<u>Field name</u>	<u>Field length</u>	<u>Offset</u>
CURRENCY_CODE	4	0
TITLE	30	4
UNITS_PLURAL	12	34
UNITS_SINGULAR	12	46

Record size = 58 bytes.

All fields are sting fields.

Note: these exercises are not optional - the file must be defined otherwise you will not be able to perform some of the exercises in later modules.

1. Create a File Definition for the CURRENCY_MASTER file.
2. Create a Field Definition for the CURRENCY_MASTER file.
3. Create a Table Definition for the CURRENCY_MASTER file.

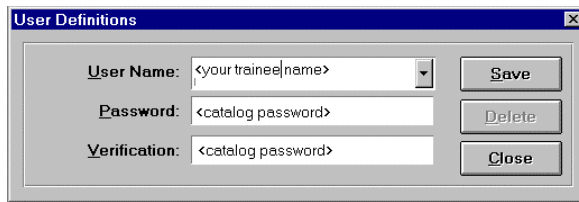
Step 7. User Definitions

1. From the Object screen select the Users button.



The current list of users is displayed; this list contains only ADMIN if no users have been previously entered.

2. Prepare to insert a new definition by either selecting **New** from the **File** pulldown menu or clicking the **New** button on the toolbar. The User Definitions dialog box appears. All the entry fields are initially blank. Enter the following information (passwords are not case sensitive).

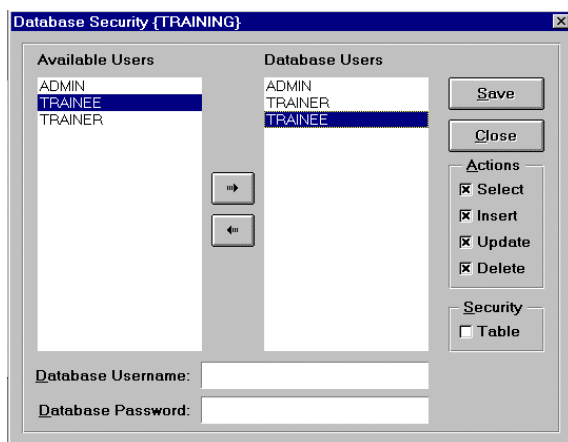


3. Press the **Save** button. After pressing the **Save** button, the Password and Verification list boxes are cleared.
4. Once the new user has been saved, **Close** the dialog box. Your user name now appears in the list of users on the Objects dialog box.


Step 8. Create Database Privileges

A user record alone is not enough to be able to access tables within the Easysoft SQL Engine. A user must be granted permissions to a particular database.

1. From the Object screen select the **Database** button. The current list of databases is displayed.
2. Double click on **TRAINING**. The Database Definition dialog box appears.
3. Click on the **Security** button. The Database Security dialog box appears. Here it is shown completed.




4. Easysoft recommend that in the general case the ADMIN user deals only with controlling the catalog and does not have access to the Server data. Therefore, this user is not given any database privileges.

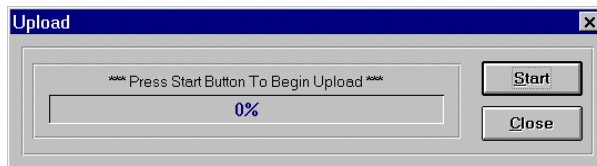
5. Highlight the *TRAINEEn* user in the Available Users list and press the  button (Add).
6. When users are added to the Database Users list they are automatically assigned all actions on the selected database. To change these actions select the database user and toggle the relevant actions. Security at table level is not automatically enabled; it is not relevant to this tutorial.

Database Username and Database Password are not applicable to Easysoft ODBC for RMS. (In applicable cases they are used to pass a name and a password to a database on the server).

7. Press the **Save** button and **Close** the dialog box.

Step 9. Upload Easysoft Catalog

1. The last action to perform is to upload the changes to the relevant Easysoft catalog on the Server.
2. Either select the **Upload** option from the **Data** pulldown menu or select the **Upload** button () on the toolbar. The Upload dialog box appears.



3. Select the **Start** button to begin the upload. The SQL Data Sources dialog box appears.
4. Select the TRAINING DATA data source.
5. Click **OK** and the Easysoft Administrator will begin the upload process to the correct Easysoft catalog directory.
6. Once the percentage bar has reached 100% the upload has completed successfully. **Close** the dialog box. The catalog on the server now contains the changes that have been made in the administrator.
7. Exit the Easysoft PC Administrator.

Exercise - connecting to the data source

- 1.** Before being able to use the TRAINING DATA data source you must change it so that it knows about the new user defined here. Do one of the following using the Easysoft ODBC Setup dialog box:
 - Make the Catalog Login Username and Catalog Login Passwords blank. When you connect to a data source you will be prompted for these by the Easysoft ODBC login prompt.
 - Set the Catalog Login Username and Catalog Login Passwords to the name and password of the user you created in step eight and save the changes.
- 2.** Connect to the TRAINING DATA data source using Access or Excel and look at the currency information contained in the two tables.

Populate Server File

In the module entitled "RMS Administration" you created a new data file called WORLD_DESTINATION.DAT on the server. In these exercises you will first define the that file in the PC Administrator and then you will populate the file from a Microsoft Access database which exists on your PC.

Here is the information that you will need for this exercise (no new information is presented here - it's given so that you don't have to flip between modules):

Filename: WORLD_DESTINATION.DAT
Record format: fixed
Record size: 43 bytes

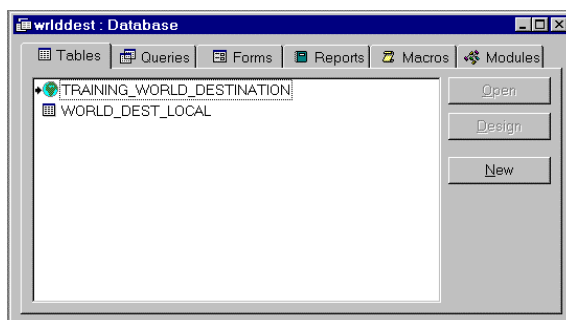
Field name	Field length (bytes)	start position (offset)	data type
WORLD_CODE	3	0	String
DESTINATION_CODE	10	2	String
TITLE	30	12	String

Exercise 1. Define the server file

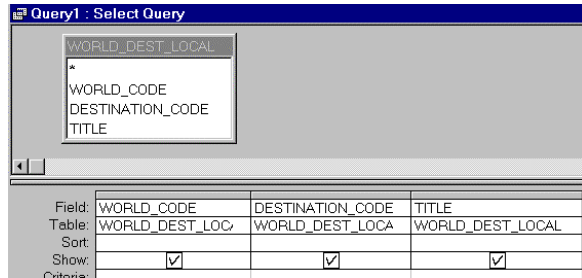
1. Create File, Field and Table Definitions for the WORLD_DESTINATION file.
2. Upload the catalog to the server and exit the PC Administrator.

Exercise 2. Upload local data to server file

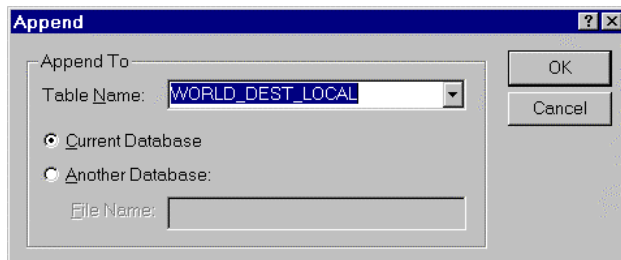
1. Start Microsoft Access and open the database which contains data to be uploaded to the server file (the name of this database will be given to you on the day of the course).
2. Create a link to the WORLD_DESTINATION table in the TRAINING DATA data source. The Access database now looks like this.



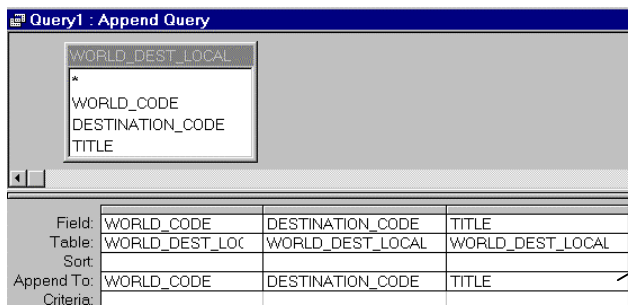
3. Start a new query (Select the Queries tab, followed by the **New** button). The New Query dialog box appears. Accept the default (Design View) and click **OK**. The Show Table dialog box appears. Add the local table containing the data (this will be given on the day of the course) to the query and close the dialog box. Add all the fields to the Query Design Grid (double-click each one in turn). The query now looks like this.




4. Append the local data to the query. Select **Query, Append...** from the main menu bar. The Append dialog box appears. Ensure that the Current Database check box is selected, and select TRAINING_WORLD_DESTINATION.



5. Click the **OK** button on the Append dialog box. The query now looks like this.



The field names are the same in both tables, so they are automatically entered here.

6. Click the Run button () to upload the data. You will be asked to confirm the upload.

Importing and Exporting Definitions


RMS data files can be described in a form that is understandable to both the Easysoft Catalog and also to human readers. This description is contained in the File and Field definitions of the Easysoft CSV (Comma Separated Values) format which is used by the Easysoft Administrator on the PC and the Easysoft Host Administrator on the server, as well as by the Easysoft catalog.

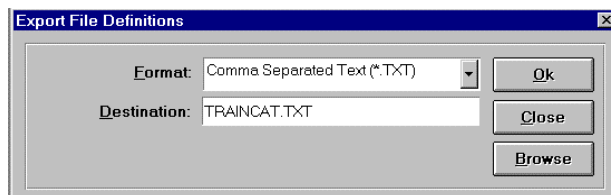
The purpose of these exercises is to show you how to import and export catalog definitions. In this set of exercises, you will export all the definitions contained in the PC Administrator to a text file, and then you will import them back into the Easysoft PC Administrator.

The default location of import/export files is C:\EASYSOFT\SQL\SYSTEM - this appears the first time in a session that the **Browse** option is used either to import or export definitions.

Exporting Definitions

In this exercise, you will export all the definitions to a CSV text file.

1. Start the Easysoft PC Administrator. Ensure you have the correct version of the catalog - download from the server if necessary.
2. Select **Export...** from the **File** pulldown menu on the main screen of the Administrator (or select the **Export** icon () from the toolbar).



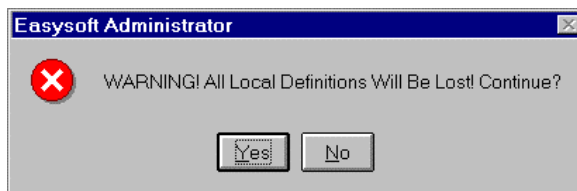
3. Ensure that the Format list box contains: Comma Separated Text (*.TXT)
4. Use the **Browse** button to locate the directory in which the file is to be stored. If you do not specify a directory, then the default directory, C:\EASYSOFT\SQL\SYSTEM, will be used. Enter the file name of the file to be written to (for example, **TRAINCAT.TXT**).
5. Click the **OK** button.
6. Verify that the file exists, use a text editor to view the contents of the file.

Note - exporting passwords


Passwords are not written on export for security reasons. Therefore, when you import the CSV text file, the passwords for User definitions (see Step 7) will be missing. If you were going to upload the catalog to the server, you would have to re-enter the user passwords before uploading the catalog. (You could also modify the CSV directly, then import it into the catalog on the server).

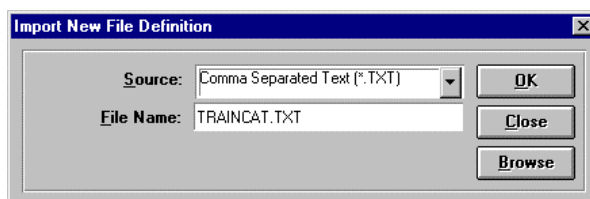
Importing Definitions

1. So that you can clearly see what is happening, first remove all the existing definitions. To do this select **Data, Clear all Definitions**.
2. You will be asked whether you want to remove all definitions. Select **Yes**.



You will notice that all definitions are removed, except for the ADMIN user definition.

3. Select **Import...** from the **File** pulldown menu on the main screen of the Administrator (or select the **Import** icon () from the toolbar). The Import New File Definition dialog box appears.



4. Ensure that the Source list box contains: Comma Separated Text
5. Use the **Browse** button to locate the file to be imported (the one you created in the previous exercise). Select the file; the name appears in the File Name list box.

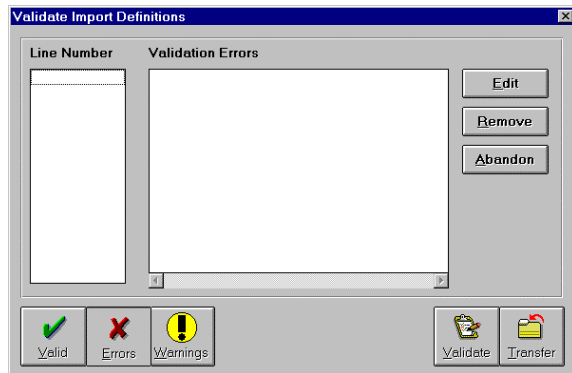
Alternatively, if the file resides in the default directory (`C:\EASYSOFT\SQL\SYSTEM`) then enter the name in the entry box.

6. Click the **OK** button.

At the bottom of the Administrator there is a percentage bar which shows how the import is progressing. There are a total of three passes to the import.

- Pass 1 import the file definitions
- Pass 2 validate the imported definition
- Pass 3 transfers validated definition into the local database, ready for transfer to the server.

7. Once a file has been imported (pass 1), the Validate Import Definitions dialog box is displayed. This is used as an aid to alter incorrectly formatted data or duplicated data within the import file (in this case, there are no errors).



See "Note - validating import definitions", page 8-26 for an example of using this dialog box.

8. Click the **Transfer** button. This transfers the imported details into the local copy of the Easysoft catalog, after which the message, `Import Completed Successfully` is shown.

Note - validating import definitions

The Validate Import Definitions dialog box is used as an aid to alter incorrectly formatted data or duplicated data within the import file. It is automatically displayed during the process of importing a definition.



This button displays those imported items which were successfully validated



This button displays those imported items which failed validation



This button displays those imported items which failed validation, but which have recoverable errors

You can switch between seeing valid lines, invalid lines and lines which have recoverable errors at any time, by selecting the appropriate button.

The items which are imported are each defined by a separate line of text. When the Validate Import Definitions dialog box first appears, it shows a list of errors and associated line numbers. If errors have occurred during import, these errors can be amended by using the **Edit** button or removed from the import by use of the **Remove** button. You are asked to confirm removal of a line. Line numbers do not change if a line is removed. An example of a typical error is that a definition already exists.

The **Validate** button should be pressed after each change to re-validate the remaining imported lines. This is necessary because correcting one line may result in the remaining lines which gave errors no longer being invalid (i.e. their validity depended on the validity of the line which was changed).

Once all lines are validated successfully or have recovered the **Transfer** button is enabled. This button transfers the imported details into the local copy of the Easysoft Catalog, after which the message, `Import Completed Successfully` is shown.

To cancel the import process at any stage prior to transferring the definitions, press the **Abandon** button.

Summary

In this module you learnt how to

- ✦ Set the password for access to the Easysoft PC Administrator
- ✦ Use the Easysoft PC Administrator to define RMS files so they can be used by ODBC-compliant applications. This included
 - downloading and uploading the catalog
 - File and Field Definitions - to describe the files on the server
 - Database Definitions - to define an SQL database
 - Table Definitions - to map SQL tables to server files
 - Column Definitions - to map SQL columns to fields in a file
 - User Definitions - to define catalog users and assign passwords
 - Database and Table Privileges - set access rights to the database and tables
- ✦ Import and export file definitions (as a CSV text file) from the Easysoft PC Administrator

Review Questions

1. Before any definitions can be entered locally using the Easysoft Administrator, what action must be performed?
2. What are the six objects for which definitions must be created using the Easysoft PC Administrator?
3. Privileges can be set for two objects; which ones are they, and which one of these is optional?
4. What must you do so that the server knows about the new catalog definitions?
5. How could you transfer definitions between different catalogs?

• Question: Why does this page contain an apparently useless question?
• Answer: to stop you taking a sneak preview of the answers (which are now overleaf)

Review Answers

1. Before any definitions can be entered locally using the Easysoft Administrator, what action must be performed?

Download catalog from server to the PC.

2. What are the six objects for which definitions must be created using the Easysoft PC Administrator?

File, Field, Database, Table, Column, User

3. Privileges can be set for two objects; which ones are they, and which one of these is optional?

Database	mandatory
Table	optional

4. What must you do so that the server knows about the new catalog definitions?

Upload the catalog to the server.

5. How could you transfer definitions between different catalogs?

Use the import and export definition options.

Field lengths and offsets (page 8-3) - Exercise answers

<u>Field name</u>	<u>Field length</u>	<u>Offset</u>
CURRENCY_CODE	4	0
TITLE	30	4
UNITS_PLURAL	12	34
UNITS_SINGULAR	12	46

Record size = 58 bytes.

Supplement: Easysoft PC Administrator Installation

This section describes the installation of the Easysoft Administrator. The installation does not require much user input; it should take no more than ten minutes to complete.

Download the Software

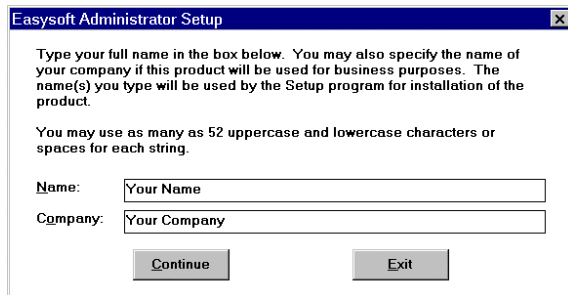
1. Create a directory in which to store the files that will be generated later from the zipped file which you are about to download. (These files can be deleted later if you wish).
2. Start a web browser.
3. Go to the Easysoft Home Page (<http://www.easysoft.com>). Navigate to the page containing the Easysoft ODBC for RMS downloadable software.
4. Click on Easysoft ODBC PC Administrator. You will be prompted to save the file (esadmin.zip). Save the file in the directory you just created.
5. Exit the web browser.
6. Unzip the file.
7. If you wish, *after* you have installed the PC Administrator, you can delete the zipped file and all the files that were generated when this was unzipped.

Administrator Installation Steps

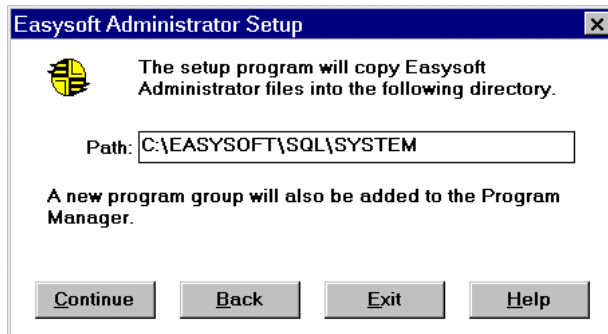
1. Close all applications.
2. Run the Setup.exe program that was generated in the unzip process.
3. After starting the installation program you first see a message: `Initialising Setup.....`
4. When the setup utility has initialised properly, the Easysoft Administrator Setup (Introductory) dialog box is displayed. Click the **Continue** button.



5. The Easysoft Administrator Setup (User Details) dialog box appears.



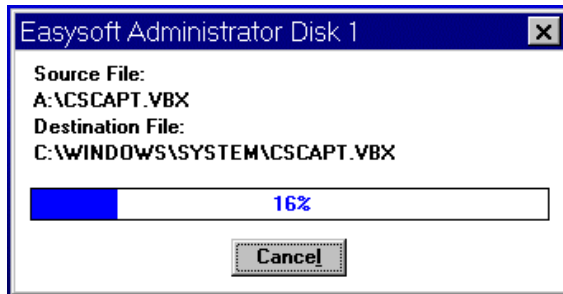
6. Enter your name and company in the appropriate list boxes. Pressing the **Enter** key on the keyboard enters the details and moves you on to the next stage, so use the mouse or tab to move the cursor from the Name entry field to the Company entry field, unless you wish the Company field to remain empty.
7. Click the **Continue** button to move to the Directory dialog box.



The Directory dialog box is used to specify the destination directory for the files which are to be installed. The default directory for the Easysoft Administrator files is C:\EASYSOFT\SQL\SYSTEM. If you do not want this to be the destination directory, enter the directory name that you require; the installation automatically creates this directory if it does not already exist.

8. Click **Continue** to begin installing the software onto your PC.

9. The File Copying dialog box appears; it indicates how the copying is progressing.



10. When the installation has completed successfully, the final dialog box is shown. Click **OK** to return to Windows.



9. Easysoft Administration on the Server

This module deals with the administration of the Easysoft system on the server platform.

In this module you will learn how to

- ✦ Create a catalog
- ✦ Change catalog passwords
- ✦ Change the Easysoft ODBC Setup dialog box to reflect changes to OpenVMS passwords
- ✦ Use Easysql to verify the SQL that is sent by an application
- ✦ Use the Easysoft COBOL converter to convert COBOL file definitions into the Easysoft CSV format

At the end of the module there are some review questions.

Contents

Dealing with Catalogs _____	9-2
Create a Catalog _____	9-2
Changing Catalog Passwords _____	9-3
Exporting a Catalog _____	9-4
Importing a Catalog _____	9-5
OpenVMS Passwords and Easysoft _____	9-6
Easysql _____	9-7
Easysoft COBOL Converter _____	9-8
Summary _____	9-9
Review Questions _____	9-10
Conversion Exercise _____	9-11
Exercise Steps _____	9-12
Review Answers _____	9-15

Dealing with Catalogs

In this section you will see how to create a catalog, change the passwords for that catalog and import and export the CSV definitions of a catalog.

Create a Catalog

A catalog will have been created during the initial installation of Easysoft ODBC for RMS. However, it is possible to have more than one catalog, and this section gives you step-by-step instructions for creating one using the Host Administrator CREATE CATALOG command.

Note: Each catalog must reside in a different directory. (Use the CREATE/DIRECTORY command to create a new directory.) A catalog must be populated before it can be used.

Start the Easysoft Host Administrator.

```
$ RUN EASYSOFT_SQL_ADMIN
```

A short message is output, and then the ADMIN> prompt appears. Run the CREATE CATALOG command. You are asked to provide the name of a directory in which the catalog is to be created.

```
ADMIN> CREATE CATALOG
_directory : <directory>
```

You can use [.] to create the catalog in the current directory.

You are then asked for a password which becomes the catalog administrator's password for this Easysoft Catalog. (The catalog administrator's username is automatically set to ADMIN).

```
_password : <enter a password here>
_retype password : <repeat the password>
```

After you have re-typed the password for confirmation purposes notification of catalog creation is presented (only the first line is shown here).

```
Creating catalog in the directory <directory>
```

Exit the Administrator by pressing the **Enter** key.

You now have a completely empty catalog.

Changing Catalog Passwords

The Catuser routine is used to change the password of existing Catalog users. It resides in the directory pointed to by the EASYSOFT_SQL_SYSTEM logical. Any catalog user can run this routine, but there is a special option for the catalog administrator. First, the general case is described, followed by the catalog administrator user options.

```
$ RUN EASYSOFT_SQL_SYSTEM:CATUSER
<informational message>
Catalog Directory : <catalog directory>
Catalog User      : <user>
Catalog Password  :
New Password      :
Verify Password   :
Catalog password successfully updated for user <user>
```

Catalog Directory refers to the directory in which the catalog resides. Either type the name of the directory (use [] for the current directory), or type a logical that points to the directory.

Catalog User refers to the user whose password you wish to change. If the specified user does not exist, an error message is generated. If the catalog administrator user is entered here, special conditions apply (see below). Catalog User equates to Catalog Login Username in the Easysoft ODBC Setup dialog box.

Type the current Catalog Password. If the password is incorrect, an error message is generated. Catalog Password equates to Catalog Login Password in the Easysoft ODBC Setup dialog box.

Enter the New Password. You must verify the new password. If the names do not match, the following message appears: `Passwords do not match`. You must re-enter both the new password and the verification.

You can quit the routine at any stage except the last one by entering a blank line. All the existing values are retained. However, if you get as far as entering a new password, then you must complete the process by entering the verification. If you enter a blank line at this stage, the verify password prompt re-appears.

Note: passwords are not case-sensitive.

Catalog Administrator Options

The catalog administrator has the option of changing passwords for all users, in addition to being able to change just their own password.

```
Catalog Directory : EASYSOFT_SQL_CATALOG
Catalog User      : ADMIN
Catalog Password  :
```

Catalog administrator's name (always "ADMIN") and password.

Administrator Options:

- 1) Change Administrator Password
- 2) Change User Password
- 0) Exit

Enter option number [1, 2 or 0] - 2

```
Catalog User      : DEMO _____ Name and new catalog password
New Password     : _____ of non-administrator user.
Verify Password  :
```

Catalog password successfully updated for user DEMO

```
Catalog User      :
```

If, at the Catalog User and Password prompts, the catalog administrator details are entered, three options are presented.

Option 1 behaves exactly as described in the previous section.

Option 2 (shown in the example above) allows the catalog administrator to change catalog passwords for any user defined in the catalog. Press the **Enter** key at the Catalog User prompt to quit the routine.

Option 3 (labelled 0) is used to quit the routine at this point.

Exporting a Catalog

To export a CSV file from the catalog into a text file, take the following steps.

Start the Easysoft Host Administrator and run the EXPORT CSV command.

```
$ RUN EASYOFT_SQL_ADMIN
ADMIN> EXPORT CSV <file to export> <catalog directory> <password>
```

<file to export> is the name of the file into which to write the CSV definitions.

<catalog directory> is the directory in which the catalog resides.

<password> is the catalog administrator's password.

When the ADMIN> prompt reappears exit from the Host Administrator by pressing the **Enter** key.

Note - passwords

Passwords are not exported, so if you do not edit the CSV before re-importing it, an error will arise when you use the software.

Importing a Catalog

To import a CSV file into the catalog on the server, take the following steps.

Start the Easysoft Host Administrator and run the IMPORT CSV command.

```
$ RUN EASYOFT_SQL_ADMIN  
ADMIN> IMPORT CSV <file to import> <catalog directory> <password>
```

```
Checking catalog version in directory <catalog directory> ...  
Catalog version <version> is valid. Import resuming ...  
<informational messages>  
Database updated successfully
```

<file to import> is the name of the file which contains the CSV definitions which are to be imported. This file must be somewhere on the server, so if, for example, you exported definitions from the PC administrator, you must copy them to the server. (Do not confuse this with the catalog upload operation which automatically transfers the catalog from the PC Administrator into the catalog on the server).

<catalog directory> is the directory in which the catalog resides.

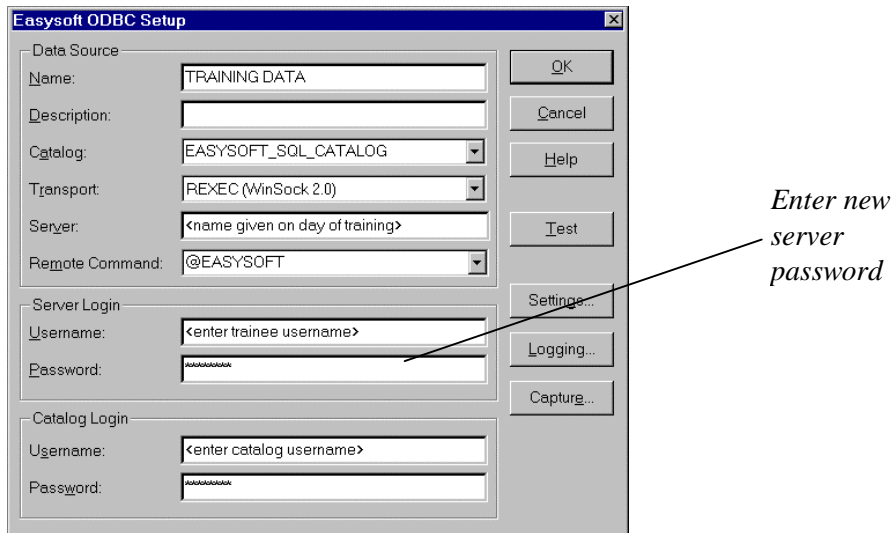
<password> is the catalog administrator's password.

When the ADMIN> prompt reappears exit from the Host Administrator by pressing the **Enter** key.

OpenVMS Passwords and Easysoft

If you change your OpenVMS password, and if the password and user information is entered in the Easysoft ODBC Setup dialog box (it need not be), then you will need to change the password.

1. Obtain the Easysoft ODBC Setup dialog box.



2. Change the password in the Easysoft ODBC Setup dialog box.
3. Press the **Test** button on the Easysoft ODBC Setup dialog box. Assuming everything is correct, you should see:



4. Make sure you save your changes by selecting **OK**.

Easysql

Easysql is a debug tool which forms part of the Easysoft Server Component. Its function is to send SQL to the server data and output the results. If problems are encountered when using an ODBC application, check whether the query that is sent from an application returns the same data as Easysql. This is what you do to use Easysql.

1. Log on to the server and set up a symbol to point to Easysql. At the command prompt type:

```
$ SQL:==$EASYSOFT_SQL_EASYSQL
```

2. Start Easysql. At the command prompt type:

```
$ SQL <catalog> <Catalog user name> <Catalog password>
```

Easysql takes three parameters

- 1) catalog directory
- 2) a Catalog user name
- 3) a password for the Catalog user

3. After a copyright message you will see the EASYSQL> prompt. At this type the SQL query:

```
EASYSQL> SELECT <columns> FROM <tables> [WHERE <conditions>];  
<result appears here>
```

The semi-colon at the end of the SQL is used to terminate the statement.

4. To exit from Easysql, press the **Enter** key at the EASYSQL> prompt.

Easysoft COBOL Converter

There are three Easysoft converters (COBOL, PowerHouse, IQ) that can be used to automatically convert file definitions into the CSV format that is used in the Easysoft catalog. This means that it is not necessary to manually input File and Table Definitions in the Easysoft PC Administrator. Here we focus on the COBOL converter, as the other two are for proprietary database systems. Full documentation for all the converters is in the Appendix.

The Easysoft COBOL converter is used to convert file definitions that are contained in the FILE SECTION of COBOL files to the CSV form required by the Easysoft catalog. The converter does not use an entire COBOL program; any text file containing valid COBOL record descriptions can be converted.

The COBOL converter is supplied as a supplement to the Easysoft software, and must be installed on your server before it can be used. The installation is quick and simple, but for convenience, the converter has already been installed for you. See the appendix for full details.

The generalised example below shows you how to generate a CSV file from a COBOL program file. (In addition to the basic conversion, you are able to output FILLER fields and Group Items. See the appendix for full details.)

1. Start the converter, and supply the name of the input file and the output file

```
$ @EASYSOFT_SQL_SYSTEM:COBOLCNV <program to convert> /OUT=<output file name>
```

```
Easysoft COBOL converter Version <version>
<copyright notice>
Processing file
Parsing COBOL data
Database Name :
```

2. After a few lines of informational output, you will be asked to supply a Database Name. This is the name that you want to call the database which holds the definitions which will be generated by the conversion. You must enter a value.

```
Database Name : <database name>
```

3. The next line, Default Directory, refers to the default location of the files stored on the server. It is optional; if a value is not entered, an empty string is generated in the output.

```
Default Directory : <database name or empty string>
```

4. A few informational messages are output, and then the conversion is complete.

Summary

In this module you learnt how to

- ✦ Create a catalog using the Easysoft Host Administrator
- ✦ Change catalog passwords using the Catuser routine
- ✦ Change the Easysoft ODBC Setup dialog box to reflect changes to OpenVMS passwords
- ✦ Use the Easysql debugging tool to retrieve server data directly
- ✦ Use the Easysoft COBOL converter to convert COBOL file definitions into the Easysoft CSV format

Review Questions

1. What exactly does the Easysoft COBOL Converter convert? And into what?
2. How can you create a catalog?
3. The catalog administrator's username is fixed. What is it?
4. Can you change the catalog administrator's username and password? How?
5. If the OpenVMS password is changed, does the Easysoft system need to be told of this?
6. How can you access server data directly using SQL without going through ODBC?

Answers on page 9-15

Conversion Exercise

In this exercise you will work through the entire process of converting a COBOL file definition such that the data files can be accessed by Easysoft ODBC. The overall steps of the exercise (detailed on the next page) are:

1. Create a new catalog
2. Convert the COBOL definitions to Easysoft CSV
3. Import the CSV into the Easysoft catalog
4. Create a new data source which uses the new catalog directory
5. Download the catalog into the Easysoft PC Administrator
6. Modify definitions in the Easysoft PC Administrator
7. Upload the catalog to the server
8. Change the catalog password
9. View the data

The file that you will convert is not a complete program - it contains just COBOL file definitions (FD section) for three files. These are CURRENCY_MASTER.DAT, CURRENCY_RATE.DAT and WORLD_DESTINATION.DAT. You have seen these files in previous modules.

The COBOL file that you will convert is shown below.

```

**
FD      CURRENCY
        VALUE OF FILE-ID "CURRENCY_MASTER.DAT".
**
1          CURRENCY.
3          CURRENCY_CODE          PICTURE IS X(4).
3          TITLE                   PICTURE IS X(30).
3          UNITS_PLURAL            PICTURE IS X(12).
3          UNITS_SINGULAR          PICTURE IS X(12).
**
FD      CURRENCY_RATE
        VALUE OF FILE-ID "CURRENCY_RATE.DAT".
**
1          CURRENCY_RATE.
3          CURRENCY_CODE          PICTURE IS X(4).
3          CURRENCY_RATE          USAGE IS COMP-2.
3          CURRENCY_DATE          PICTURE IS X(11) VALUE IS "01-JAN-
1997".
**
FD      WORLD_DESTINATION
        VALUE OF FILE-ID "WORLD_DESTINATION.DAT".
**
1          WORLD_DESTINATION.
3          WORLD_CODE             PICTURE IS X(3).
3          DESTINATION_CODE       PICTURE IS X(10).
3          TITLE                   PICTURE IS X(30).

```

For your convenience, a number of logicals have been defined. They are:

EASYSOFT_SQL_DATA	directory containing the training data
EASYSOFT_SQL_COBOL_DATA	directory containing COBOL text file to convert
EASYSOFT_SQL_COBOL_CATALOG	empty directory where you will create a catalog

Exercise Steps

1. Create a new catalog

Create a New Catalog on the server using the Easysoft Host Administrator CREATE CATALOG command (see "Create a Catalog", page 9-2 for details).

For your convenience, an empty directory already exists, and a logical has been created which points to this directory. At the `_directory` prompt use:
EASYSOFT_SQL_COBOL_CATALOG

Let's you forget, write the password that you use here: _____

2. Convert the COBOL definitions to Easysoft CSV

a) First set the default directory to be the one where the COBOL text file resides
\$ SET DEF EASYSOFT_SQL_COBOL_DATA

b) Then view the contents of the COBOL file (it should look like the file on the previous page)
\$ TYPE COBOL.TXT

c) Next convert the COBOL definitions to Easysoft CSV.
\$ @EASYSOFT_SQL_SYSTEM:COBOLCNV COBOL.TXT /OUT=COBOL.CSV

When prompted, enter the following

Database Name: COBOL

Default Directory: EASYSOFT_SQL_DATA

*This is where the
RMS data files reside*

d) Finally look at the CSV - there is no user information.
\$ TYPE COBOL.CSV

3. Import the CSV into the Easysoft catalog

Import the CSV into the Easysoft catalog on the server (see "Importing a Catalog", page 9-5).

a) Start the Easysoft Host Administrator
\$ RUN EASYSOFT_SQL_ADMIN

b) Run the IMPORT CSV command.
ADMIN> IMPORT CSV COBOL.CSV EASYSOFT_SQL_COBOL_CATALOG ADMIN
<pwd>

<pwd> is the password you used in step 1.



4. Create a new data source

Create a new data source which uses the new catalog. Use the Microsoft ODBC Administrator to obtain the Easysoft ODBC Setup dialog box (see module entitled "Microsoft ODBC Administrator" for full details). Enter the following information:

Data Source Name: **COBOL**

Catalog: **EASYSOFT_SQL_COBOL_CATALOG**

Transport: **REXEC**

Server: **MODI**

Server Login Username and password: your trainee username and password as shown on the front of your PC

Catalog Login: **ADMIN**

Catalog Password: the password you used in step 1



5. Download the catalog into the PC Administrator

Download the catalog into the Easysoft PC Administrator (ensure you connect to the COBOL data source, not the TRAINING DATA data source - they use different catalogs).

(Refer to "Step 1. Download Easysoft Catalog" in the module entitled "Easysoft Administration on the PC" for full details).



6. Modify definitions in the Easysoft PC Administrator

- a) Using the Field Details dialog box, change the datatype of the CURRENCY_RATE file from STRING to DATE. Don't forget to save the change.

(Refer to "Step 3. Field Definitions" in the module entitled "Easysoft Administration on the PC" for full details).

- b) Add a new user - use a name and password of your choice.

User name: _____ Password: _____

(Refer to "Step 7. User Definitions" in the module entitled "Easysoft Administration on the PC" for full details).

- c) Give the new user access rights to the database.

(Refer to "Step 8. Create Database Privileges" in the module entitled "Easysoft Administration on the PC" for full details).

7. Upload the catalog to the server

Upload the catalog to the server, then exit the Easysoft PC Administrator.

Note: Only the ADMIN user has write privileges to the catalog. Therefore you must ensure that the data source uses the ADMIN user for the Catalog Login option.

(Refer to "Step 9. Upload Easysoft Catalog" in the module entitled "Easysoft Administration on the PC" for full details).

8. Change the catalog password

- a) Use the Catuscr routine on the server to change the catalog password. Refer to "Changing Catalog Passwords", page 9-3, for full details.

Write the new password here: _____

- b) Update the data source to reflect this new password and then test the connection (not necessary for the remainder of this exercise).

9. View the data

- a) Run Easysql on the server to view the data (refer to "Easysql", page 9-7, for details. You may wish to use this example query:

```
EASYSQL> SELECT * FROM COBOL_WORLD_DESTINATION;
```

- b) You may also like to view the data using Easysoft Query for Windows on the PC (or any other ODBC-compliant PC application).

Review Answers

1. What exactly does the Easysoft COBOL Converter convert? And into what?

It converts File Definitions contained in the FD section of a COBOL file (it can also convert definitions contained in a text file). These definitions are converted into the equivalent CSV format which can be imported into the Easysoft catalog.

2. How can you create a catalog?

Use the CREATE CATALOG command in the Easysoft Host Administrator.

3. The catalog administrator's username is fixed. What is it?

ADMIN.

4. Can you change the catalog administrator's username and password? How?

Username cannot be changed.
Password can be changed using the Catuser routine

5. If the OpenVMS password is changed, does the Easysoft system need to be told of this?

Yes. The Server Login Password in the Easysoft ODBC Setup dialog box (accessed using Microsoft ODBC Administrator on the PC) needs to be changed.

6. How can you access server data directly using SQL without going through ODBC?

Use the Easysql debug utility.

10. RMS Training Data

This module describes the Easysoft Travel Company which is used as the basis of the examples in this course. There is nothing to learn; the contents of this module are for reference purposes when you do the practical exercises.

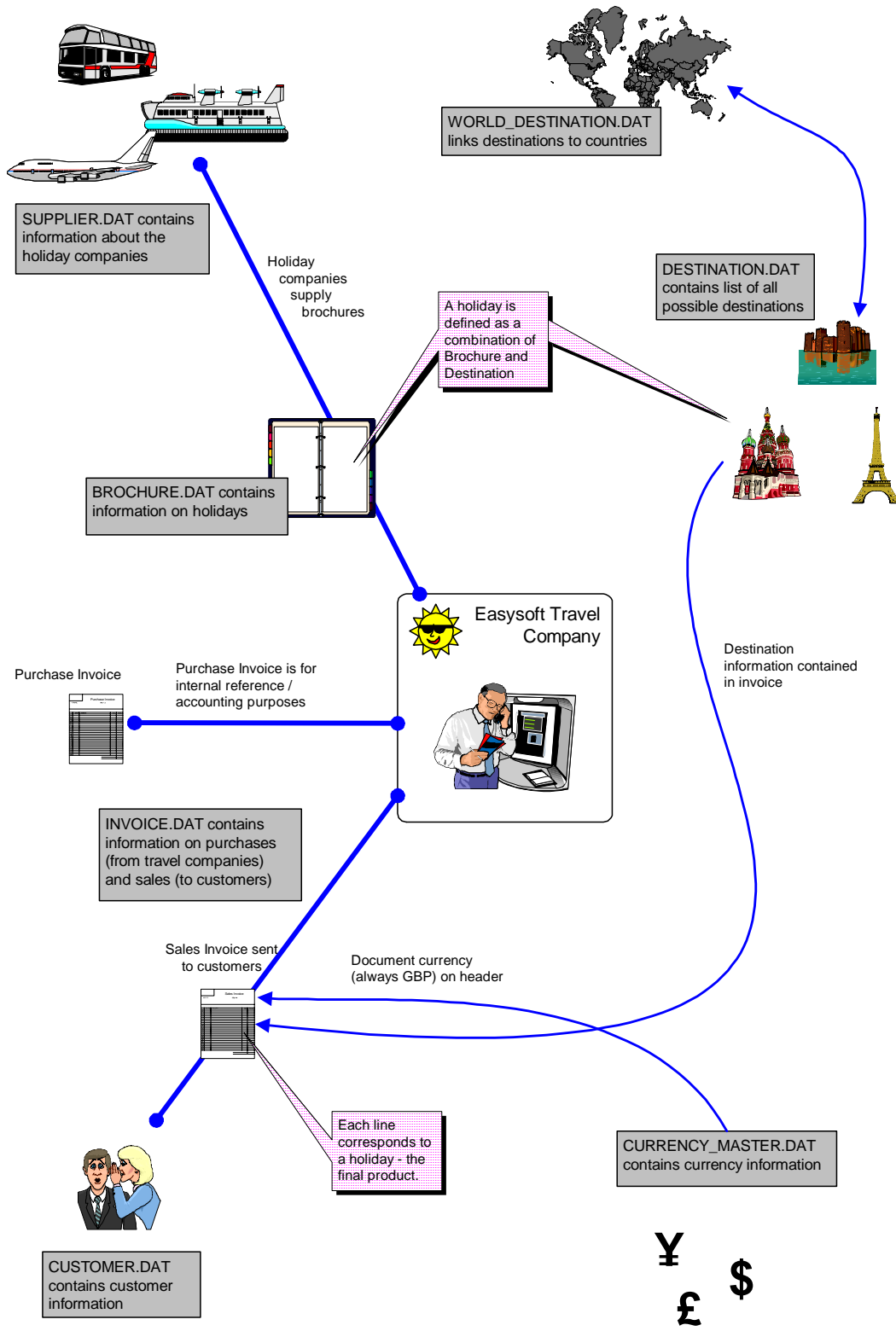
In this module you will see

- ✦ an overview of the training company
- ✦ its overall structure
- ✦ FDL descriptions of the RMS files
- ✦ some of the data contained in the files
- ✦ CSV descriptions of the Easysoft catalog

Contents

Easysoft Travel Company _____	10-2
Description of RMS Data Files _____	10-3
SQL Tables _____	10-5
RMS Training Data _____	10-6
FDLs for RMS Files _____	10-12
CSV Description _____	10-21

Easysoft Travel Company



The example company that we use is a small independent travel agency which is based in the city of Leeds.

Holidays are purchased from various tour operators, and these are then sold to the public. This example focuses on the customers, suppliers, currencies and holidays that are bought and sold. The diagram on the previous page indicates the overall structure of the system.

The first section of this module describes the RMS data files in conceptual terms, then the SQL tables are described. The data for customers, suppliers, destinations and brochures is then listed. The RMS files are then described in detail using FDLs. Finally, the Easysoft CSV for the catalog is listed.

Description of RMS Data Files

This section describes the content of the RMS data files; a technical description of the structure of these can be found in “FDLs for RMS Files” on page 10-12.

BROCHURE.DAT

This contains a list of all the brochures that are supplied by the holiday companies. Each brochure has a unique identifier - the brochure code. A brochure code has the following structure: *Bnnnn*, where *n* represents a single digit.

CURRENCY_MASTER.DAT

This contains a list of all currencies available to the system. The records in the file contain the following information: currency code, title, name of plural units, name of singular units. Each currency has a unique identifier - the currency code which consists of three characters e.g. USD, GBP.

Document Currency is an accounting term which refers to the unit of currency of a document (e.g. sales invoice, purchase order). Sales and purchase invoices in the Easysoft Travel Company have a field called Document Currency. In all cases, this contains the value GBP (i.e. Pounds Sterling). This field has nothing to do with currency exchange transactions that a customer may perform. The currency files has been included in the system since they are a real-world example of files that have relatively few fields, and are therefore convenient to use for many of the hands-on examples used in this training course.

CURRENCY_RATE.DAT

This file contains exchange rates with respect to Sterling.

CUSTOMER.DAT

This contains a list of all customers. Each customer has a unique identifier - the customer code. A customer code has the following structure: *Cnnnn*, where *n* represents a single digit.

DESTINATION.DAT

This contains a list of all destinations known to the system. Each destination has a unique identifier - the destination code. A destination code has the following structure: *Dnnnn*, where *n* represents a single digit.

INVOICE.DAT

This contains information on all sales to customers and purchases from holiday companies. Each sale or purchase is physically recorded on a Sales Invoice or a Purchase Invoice. The information shown on these invoices is recorded in the INVOICE.DAT file. Each physical invoice has a unique identifier - the invoice number.

The invoice types are distinguished in the data file by a field called TYPE. Sales invoices are identified by the value SINV and purchase invoices are identified by the value PINV.

Physically, invoices have one header line and one or more details lines (see “Notes” on page 10-5). From the perspective of customers, the brochures contain all the information necessary to purchase a holiday to a particular destination. However, from an internal perspective, the holidays are defined on invoices; each detail line represents a holiday to a particular destination.

SUPPLIER.DAT

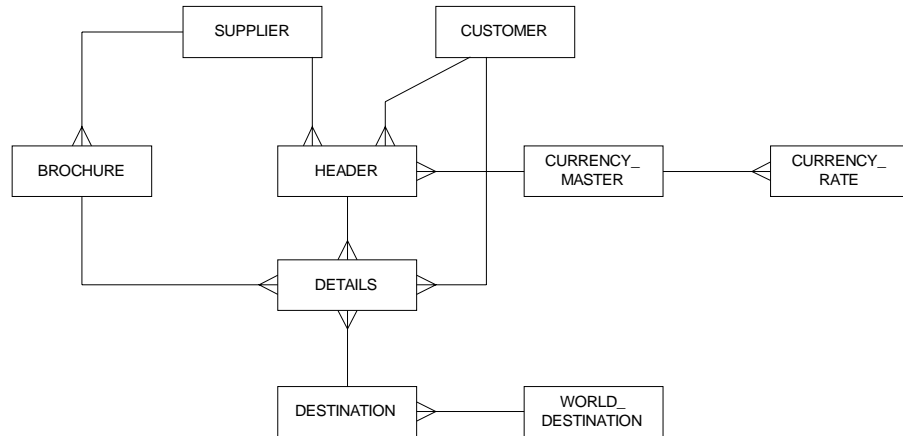
This contains a list of all suppliers, their addresses and contact information. Each supplier has a unique identifier - the supplier code. A supplier code has the following structure: *Snnnn*, where *n* represents a single digit.

WORLD_DESTINATION.DAT

This contains a list of all destinations and the corresponding country code for each destination (the country code is equivalent to the currency code in the CURRENCY_MASTER file).

SQL Tables

The SQL table structure that is defined is shown below.



The SQL tables map on to the RMS files as shown below.

Correspondences between RMS files and SQL tables	
RMS FILE	SQL TABLE
BROCHURE.DAT	BROCHURE
CURRENCY_MASTER.DAT	CURRENCY_MASTER
CURRENCY_RATE.DAT	CURRENCY_RATE
CUSTOMER.DAT	CUSTOMER
DESTINATION.DAT	DESTINATION
INVOICE.DAT	HEADER, DETAILS
SUPPLIER.DAT	SUPPLIER
WORLD_DESTINATION.DAT	WORLD_DESTINATION

Notes

If the LINE_NUMBER field in a record in the INVOICE.DAT file has the value 0, then it appears in the HEADER table.

If the LINE_NUMBER field in a record in the INVOICE.DAT file has the value 1 or greater, then it appears in the DETAIL table.

RMS Training Data

List of Customers

CUSTOMER_CODE	NAME	FORENAME	TITLE	ADDRESS1	ADDRESS2	ADDRESS3	TOWN	COUNTY	POSTCODE	TELEPHONE	FAX	COMMENTS
C0001	Richardson	Carolyn	Miss	Raths Rygg	Woodhouse Lane	Hove Edge	Brighouse	West Yorkshire	HD6 4JR	01484 719081		Good customer
C0002	Kos	John	Mr	Stone Cut Cottage	Cluff Road		Wakefield	West Yorkshire	WF11 9JF	01977 677899		USA every Xmas
C0003	Crummack	Jason	Mr	66 Collingway Close			Pontefract	West Yorkshire	WF7 6HT	01977 782445		
C0004	Welburn	Caroline	Miss	111 Altof Street			Wakefield	West Yorkshire	WF5 2JH	01924 222088		Own apartment in TFS
C0005	Smith	Guy	Mr	33 Barley Road			Wetherby	West Yorkshire	LS23 8PW	01937 566132		
C0006	Berlin	Karl	Mr	Top Flat	32 Moore Croft Lane		Leeds	West Yorkshire	LS6 4BJ			No phone
C0007	Gothard	Keely	Miss	44 Park Road West	Sherburn		Leeds	West Yorkshire	LS22 6JF	01977 688764		
C0008	Naylor	Ian	Mr	99 Bolton Lane			Rochdale	Lancashire	OL31 1NJ	01706 855606		
C0009	Gorman	Nick	Mr	78 Triangle Walk			Halifax	West Yorkshire	HX6 7GH	01422 886792		
C0010	Unwalla	Mike	Mr	224 Spoony Lane			Sheffield	West Yorkshire	S11 6BJ	0114 2997723		Long Haul customer
C0011	Smith	Martin	Mr	38 Watergate Way	Harehills		Leeds	West Yorkshire	LS3 3BJ	0113 2220600		
C0012	Carter	Michael	Mr	Flat49	The Calls		Leeds	West Yorkshire	LS1 7HH	0113 2221769		
C0013	Gillard	Simon	Mr	Fresh Fields	Moor Road		Halifax	West Yorkshire	HX2 6YH	01442 298700		
C0014	Chan	Tony	Mr	Walklane Lane	Horsforth		Leeds	West Yorkshire	LS1 7YH	0113 5678999		
C0015	Spencer	Richard	Mr	33 Manningham Lane			Bradford	West Yorkshire	BD6 9NU	01274 677879		
C0016	Beeby	John	Mr	3 Town Street	Horsforth		Leeds	West Yorkshire	LS4 4DU	01132260089		
C0017	Morley	Mark	Mr	1 Street Way	ChapelTown		Leeds	West Yorkshire	LS6 1RT	01132 867772		
C0018	Jones	Alison	Miss	33 Church street			Yeadon	West Yorkshire	LS23 5TT	01132 659980		
C0019	Cornwell	Richard	Mr	23 Bramston Lane			Brighouse	West Yorkshire	HD5 6RT	01484 722907		
C0020	Crowther	Louise	Mrs	103 Bradford Road			Bradford	West Yorkshire	BD9 9NU	01274 675589		

List of Brochures

BROCHURE_CODE	SUPPLIER_CODE	DESCRIPTION
B0001	S0001	Faraway Shores
B0002	S0001	Ski-ing
B0003	S0001	Young At Heart
B0004	S0001	Price Breakers
B0005	S0001	City Breaks
B0006	S0001	World Wide
B0007	S0001	Flight Only
B0008	S0001	A La Carte
B0009	S0001	Weddings In Paradise
B0010	S0001	All Inclusive
B0011	S0001	Summer Sun
B0012	S0001	Winter Sun
B0013	S0002	Airtours Summer Sun
B0014	S0002	Airtours Winter Sun
B0015	S0002	Airtours Far and Away
B0016	S0002	Weddings of Fantasy
B0017	S0002	All Inclusive
B0018	S0002	Airtours Flight Only
B0019	S0002	Airtours Florida
B0020	S0002	Airtours Cruises
B0021	S0002	Golden Years
B0022	S0002	Airtours Fly-Drive
B0023	S0002	Distant Dream
B0024	S0003	Dream Weddings
B0025	S0003	Cosmos Sandals
B0026	S0003	Cosmos Summer Sun
B0027	S0003	Cosmos Winter Sun
B0028	S0004	Kuoni World Wide
B0029	S0004	Australia & Newzealand
B0030	S0004	Caribbean & All Inclusive
B0031	S0004	Kuoni Switzerland
B0032	S0004	Kuoni Cruise & Stay
B0033	S0004	Kuoni Safaris
B0034	S0004	Maldiv & Sri Lanka
B0035	S0004	Kuoni Weddings
B0036	S0004	Kuoni South America
B0037	S0004	Kuoni Three
B0038	S0004	America & Canada

BROCHURE_CODE	SUPPLIER_CODE	DESCRIPTION
B0039	S0004	Ski Switzerland
B0040	S0005	Cresta Cities
B0041	S0005	Cresta Rome
B0042	S0005	Cresta Italy
B0043	S0005	Cresta Ireland
B0044	S0005	Cresta France
B0045	S0005	Disneyland Paris
B0046	S0005	Cresta Eurostar
B0047	S0005	Golf In France
B0048	S0006	Beaches
B0049	S0006	Italian Cities
B0050	S0006	Italian Lakes
B0051	S0006	Italian Countryside
B0052	S0006	Italian Islands
B0053	S0007	Getaway Breaks
B0054	S0007	Christmas & New Year
B0055	S0007	Footloose
B0056	S0007	Mexico & USA W/Sun
B0057	S0007	Trek America
B0058	S0008	Live the Dream
B0059	S0008	Hawaii
B0060	S0008	Las Vegas
B0061	S0008	Tennessee
B0062	S0009	Destination New England
B0063	S0010	Fly-Drive
B0064	S0010	Coach Tours
B0066	S0010	Motor Homes
B0067	S0010	Hotels & Apartments
B0068	S0011	Bales Egypt
B0069	S0011	Bales WorldWide
B0070	S0012	Florida & Caribbean
B0071	S0012	Weddings
B0072	S0012	Fly-Drive
B0073	S0012	Flight Only
B0074	S0013	P&O Hovercraft
B0075	S0013	P&O Ferries
B0076	S0013	P&O Cruises

List of Currencies

CURRENCY_CODE	TITLE
AED	United Arab Emirate Dirham
ALL	Albanian Lek
ATS	Austrian Schilling
AUD	Australian Dollar
BDT	Bangladeshi Taka
BEF	Belgian Franc
BGL	Bulgarian Lev
BHD	Bahraini Dinarling
BIF	Burundi Franc
BRL	Brazilian Real
BWP	Botswana Pula
CAD	Canadian Dollar
CHF	Swiss Franc
CLP	Chilean Peso
CNY	Yuan (Chinese) Renminbi
COP	Colombian Peso
CSK	Czech Koruna
CVE	Cape Verde Escudo
CYP	Cyprus Pound
DEM	German Mark
DKK	Danish Krone
DZD	Algerian Dinar
ECS	Ecuador Sucre
EEK	Estonian Krone
EGP	Egyptian Pound
ESP	Spanish Peseta
FIM	Finnish Markka
FJD	Fiji Dollar
FRF	French Franc
GBP	British Pound
GHC	Ghanaian Cedi
GMD	Gambian Dalasi

CURRENCY_CODE	TITLE
GRD	Greek Drachma
HKD	Hong Kong Dollar
HRK	Croatian Kuna
HUF	Hungarian Forint
IDR	Indonesian Rupiah
IEP	Irish Punt
ILS	Israeli New Shekel
INR	Indian Rupee
IRR	Iranian Rial
ISK	Iceland Krona
ITL	Italian Lire
JOD	Jordan Dinar
JPY	Japanese Yen
KES	Kenyan Shilling
KRW	Korean Won
KZT	Kazakhstan Tenge
LBP	Lebanese Pound
LKR	Sri Lanka Rupee
LSL	Lesotho Loti
LTL	Lithuanian Lita
LVL	Latvian Lat
MAD	Moroccan Dirham
MRO	Mauritanian Ouguiya
MTL	Maltese Lira
MUR	Mauritius Rupee
MWK	Malawi Kwacha
MXP	Mexican Peso
MYR	Malaysian Ringgit
MZM	Mozambique Metical
NAD	Namibian Dollar
NGN	Nigerian Naira
NLG	Dutch Guilder

CURRENCY_CODE	TITLE
NOK	Norwegian Kroner
NZD	New Zealand Dollar
OMR	Omani Rial
PEN	Peruvian Nuevo Sol
PHP	Philippine Peso
PLZ	Polish Zloty
PTE	Portuguese Escudo
PYG	Paraguay Guarani
QAR	Qatari Rial
ROL	Romanian Leu
SAR	Saudi Riyal
SBD	Solomon Islands Dollar
SCR	Seychelles Rupee
SEK	Swedish Krona
SGD	Singapore Dollar
SIT	Slovenian Tolar
SKK	Slovak Koruna
SUR	Russian Rouble
SZL	Swaziland Lilangeni
THB	Thai Bhat
TRL	Turkish Lire
TWD	Taiwan Dollar
UAK	Ukraine Karbovanets
UGS	Uganda Schilling
USD	US Dollar
VEB	Venezuelan Bolivar
VUV	Vanuatu Vatu
XAF	CFA Franc BEAC
XEU	ECU
XPD	Palladium
XQU	South African Rand
ZWD	Zimbabwe Dollar

List of Destinations

DESTINATION_CODE	DESCRIPTION
D0001	London
D0002	Leeds
D0003	Edinburgh
D0004	Blackpool
D0005	Paris
D0006	Cannes
D0007	Calais
D0008	Bordeaux
D0009	Chamonix
D0010	Rome
D0011	San Remo
D0012	Venice
D0013	Sorrento
D0014	Lake Como
D0015	Lake Maggiore
D0016	Benidorm
D0017	Barcelona
D0018	Benalmedina
D0019	Torremolinos
D0020	Andorra
D0021	Magaluf
D0022	Playa De Las Americas
D0023	Brighton
D0024	Las Palmas
D0025	Gibraltar
D0026	Tunis
D0027	Banjul (Gambia)
D0028	Amsterdam
D0029	Athens
D0030	Berlin
D0031	Bruges
D0032	Boston (USA)
D0033	Budapest (Hungary)
D0034	Chicago (USA)
D0035	Cologne (Germany)

DESTINATION_CODE	DESCRIPTION
D0036	Brussels
D0037	Copenhagen
D0038	Cork - Ireland
D0039	Dubai
D0040	Dublin
D0041	Geneva (Switzerland)
D0042	Granada
D0043	Guernsey
D0044	Helsinki (Finland)
D0045	Istanbul
D0046	Kusadasi (Turkey)
D0047	The Algarve
D0048	Lisbon
D0049	Lucerne
D0050	Lyon
D0051	Madeira
D0052	Madrid
D0053	Malta
D0054	Marrakech (Morocco)
D0055	Milan
D0056	Monte Carlo
D0057	Montreal (Canada)
D0058	Munich
D0059	New York
D0060	Nice
D0061	Oslo (Norway)
D0062	Prague
D0063	Reykjavik (Iceland)
D0064	Rio De Janeiro
D0065	Salamanca
D0066	Salzburg
D0067	Seville
D0068	Stockholm (Sweden)
D0069	Toronto
D0070	Toulouse

DESTINATION_CODE	DESCRIPTION
D0071	Vienna
D0072	Warsaw
D0073	Washington D.C.
D0074	Daytona
D0075	Orlando
D0076	Tampa
D0077	Clearwater Bay
D0078	Miami
D0079	Las Vegas
D0080	Denver Colorado
D0081	Sierra Nevada
D0082	New Orleans
D0083	Cancun
D0084	Playa Dorada
D0086	Montego Bay
D0087	St Lucia
D0088	England Harbour Antigua
D0089	Honolulu
D0090	Luxor
D0091	Ari Atol (Maldives)
D0092	Phuket (Thailand)
D0093	Goa
D0094	Columbo
D0095	Cairnes
D0096	Sydney
D0097	Auckland
D0098	Beijing
D0099	Hong Kong
D0100	Grand Baie
D0101	Mombassa
D0102	Cape Town
D0103	Trinidad (CUBA)
D0104	Bermuda
D0105	Paradise Island
D0106	Nerja

List of Suppliers

SUPPLIER_CODE	NAME	ADDRESS1	ADDRESS2	ADDRESS3	TOWN	COUNTY	POSTCODE	PHONE	FAX	COMMENTS	CONTACT_NAME	CONTACT_TITLE	CONTACT_FORENAME	CONTACT_PHONE	CONTACT_EXTEN
S0001	Thomson Holidays Ltd	Greater London House	Hamstead Road		London	Greater London	WC1 7SD	0990 502399	0990 502590	10% commission	Collins	Miss	Sarah		500
S0002	Airtous Holidays Ltd	Wavell House	Holcombe Road		Helmshore	Lancashire	BB4 4BN	01706 420000	01706 250000	Group discounts	Readshaw	Miss	Jane		2344
S0003	Cosmosair Plc	Tourama House	17 Homesdale Road		Bromley	Kent	BR2 9LX	0181 3131941			Sutcliffe	Mr	Mark		
S0004	Kuoni Travel	5 Exchange Street			Manchester	Greater Manchester	M2 7HA	0161 8320667							
S0005	Cresta Holidays	Tabley Court	Victoria Street		Altoncham	Cheshire	WA14 1EZ	0161 9290000	0161 9280000		Smith	Miss	Justine		
S0006	Majic of Italy	227 Shepherds Bush Road			London	Greater London	WC6 7AS	0181 748 7575		Free Children					
S0007	Best Western	Vine House	143 London Road		Kingston Upon Thames	Surrey	KT2 6NA	0181 5471515							
S0008	Trek America	4 Waterperry Court	Middleton Road	Banbury	Oxon	Oxfordshire	OX16 8QG								
S0009	North American Travel Service	Suite 2000	11 Albion Street		Leeds	West Yorkshire	LS1 5ER	0113 2461466	0113 2440481						
S0010	American Airlines Holidays Ltd	Trinity Square	23-9 Staines Road		Hunslowe	Middlesex	TW3 3HE	0181 5779966							
S0011	Bales Tours Ltd	Bales House	Junction Road		Dorking	Surrey	RH4 3HL	01306 885991							
S0012	Virgin Holidays Ltd	The Galleria	Ground Floor	Station Road	Crawley	West Sussex	RH1 1WW	01293 617181			Branson	Mr	Richard	01803 291123	
S0013	P&O Holidays	77 New Oxford Street			London	Greater London	WC1 1PP	0171 8002222		Weekend offers	Casson	Mrs	Lindy		

List of World Destinations

WORLD_CODE	DESTINATION_CODE	TITLE
AED	D0039	United Arab Emirates
ATS	D0066	Austria
ATS	D0071	Austria
AUD	D0095	Australia
AUD	D0096	Australia
AUD	D0097	Australia
BEF	D0031	Belgium
BEF	D0036	Belgium
BRL	D0064	Brazil
CAD	D0057	Canada
CAD	D0069	Canada
CHF	D0041	Switzerland
CNY	D0098	China
CSK	D0062	Czech
DEM	D0030	Germany
DEM	D0035	Germany
DEM	D0058	Germany
DKK	D0037	Denmark
EGP	D0090	Egypt
ESP	D0016	Spain
ESP	D0017	Spain
ESP	D0018	Spain
ESP	D0019	Spain
ESP	D0020	Spain
ESP	D0021	Spain
ESP	D0022	Spain
ESP	D0024	Spain
ESP	D0025	Spain
ESP	D0042	Spain
ESP	D0051	Spain
ESP	D0052	Spain
ESP	D0065	Spain
ESP	D0067	Spain
ESP	D0106	Spain
FIM	D0044	Finland

WORLD_CODE	DESTINATION_CODE	TITLE
FRF	D0005	France
FRF	D0006	France
FRF	D0007	France
FRF	D0008	France
FRF	D0009	France
FRF	D0050	France
FRF	D0056	France
FRF	D0060	France
FRF	D0070	France
GBP	D0001	Britain
GBP	D0002	Britain
GBP	D0003	Britain
GBP	D0004	Britain
GBP	D0023	Britain
GBP	D0025	Britain
GBP	D0043	Britain
GMD	D0027	Gambia
GRD	D0029	Greece
HKD	D0099	Hong Kong
HUF	D0033	Hungary
IEP	D0038	Ireland
IEP	D0040	Ireland
INR	D0093	India
ISK	D0063	Iceland
ITL	D0010	Italy
ITL	D0011	Italy
ITL	D0012	Italy
ITL	D0013	Italy
ITL	D0014	Italy
ITL	D0015	Italy
ITL	D0049	Italy
ITL	D0055	Italy
KES	D0101	Kenya
LKR	D0094	Sri Lanka
MAD	D0054	Morocco

WORLD_CODE	DESTINATION_CODE	TITLE
MTL	D0053	Malta
MUR	D0100	Mauritius
MXP	D0083	Mexico
MXP	D0084	Mexico
MYR	D0091	Malaysia
NLG	D0028	Dutchland
NOK	D0061	Norway
PLZ	D0072	Poland
PTE	D0047	Portugal
PTE	D0048	Portugal
SEK	D0068	Sweden
THB	D0092	Thailand
TRL	D0046	Turkey
USD	D0032	USA
USD	D0034	USA
USD	D0059	USA
USD	D0073	USA
USD	D0074	USA
USD	D0075	USA
USD	D0076	USA
USD	D0077	USA
USD	D0078	USA
USD	D0079	USA
USD	D0080	USA
USD	D0081	USA
USD	D0082	USA
USD	D0086	USA
USD	D0087	USA
USD	D0088	USA
USD	D0089	USA
USD	D0103	USA
USD	D0104	USA
USD	D0105	USA
XQU	D0102	South Africa

FDLs for RMS Files

The RMS structure of the data files is shown using the FDLs for these files.

BROCHURE.FDL

```

TITLE   "Brochure"
IDENT   "22-MAY-1997 10:08:30  OpenVMS FDL Editor"
SYSTEM
SOURCE   "OpenVMS"

FILE
NAME     "BROCHURE.DAT"
ORGANIZATION  indexed

RECORD
CARRIAGE_CONTROL  carriage_return
FORMAT            fixed
SIZE              275

AREA 0
ALLOCATION          156
BEST_TRY_CONTIGUOUS  yes
BUCKET_SIZE        3
EXTENSION          39

AREA 1
ALLOCATION          6
BEST_TRY_CONTIGUOUS  yes
BUCKET_SIZE        3
EXTENSION          3

AREA 2
ALLOCATION          21
BEST_TRY_CONTIGUOUS  yes
BUCKET_SIZE        3
EXTENSION          6

KEY 0
CHANGES           no
DATA_AREA          0
DATA_FILL          100
DATA_KEY_COMPRESSION  yes
DATA_RECORD_COMPRESSION  yes
DUPLICATES         no
INDEX_AREA         1
INDEX_COMPRESSION  no
INDEX_FILL         100
LEVEL1_INDEX_AREA  1
NAME               "BROCHURE_CODE"
PROLOG             3
SEG0_LENGTH        10
SEG0_POSITION      0
TYPE               string

KEY 1
CHANGES           yes
DATA_AREA          2
DATA_FILL          100
DATA_KEY_COMPRESSION  yes
DUPLICATES         yes
INDEX_AREA         2
INDEX_COMPRESSION  no
INDEX_FILL         100

```

LEVEL1_INDEX_AREA	2
NAME	"SUPPLIER_CODE"
SEG0_LENGTH	10
SEG0_POSITION	10
TYPE	string

CURRENCY_MASTER.FDL

TITLE	"CURRENCY_MASTER"
IDENT	" 9-JUL-1997 15:03:42 OpenVMS FDL Editor"
SYSTEM	
SOURCE	"OpenVMS"
FILE	
NAME	"DKA300:[TRAIN_RMS.DATA] CURRENCY_MASTER.DAT"
ORGANIZATION	indexed
RECORD	
CARRIAGE_CONTROL	carriage_return
FORMAT	fixed
SIZE	58
AREA 0	
ALLOCATION	51
BEST_TRY_CONTIGUOUS	yes
BUCKET_SIZE	3
EXTENSION	12
AREA 1	
ALLOCATION	6
BEST_TRY_CONTIGUOUS	yes
BUCKET_SIZE	3
EXTENSION	3
KEY 0	
CHANGES	no
DATA_AREA	0
DATA_FILL	100
DATA_KEY_COMPRESSION	yes
DATA_RECORD_COMPRESSION	yes
DUPLICATES	no
INDEX_AREA	1
INDEX_COMPRESSION	no
INDEX_FILL	100
LEVEL1_INDEX_AREA	1
NAME	"CURRENCY_CODE"
PROLOG	3
SEG0_LENGTH	4
SEG0_POSITION	0
TYPE	string

CURRENCY_RATE.FDL

TITLE	"CURRENCY_RATE"
IDENT	"14-JUL-1997 11:25:22 OpenVMS FDL Editor"
SYSTEM	
SOURCE	"OpenVMS"
FILE	
NAME	"DKA300:[TRAIN_RMS.DATA]CURRENCY_RATE.DAT"
ORGANIZATION	indexed
RECORD	
CARRIAGE_CONTROL	carriage_return
FORMAT	fixed
SIZE	23

```

AREA 0
  ALLOCATION          30
  BEST_TRY_CONTIGUOUS  yes
  BUCKET_SIZE       3
  EXTENSION         6

AREA 1
  ALLOCATION          6
  BEST_TRY_CONTIGUOUS  yes
  BUCKET_SIZE       3
  EXTENSION         3

KEY 0
  CHANGES           no
  DATA_AREA         0
  DATA_FILL        100
  DATA_KEY_COMPRESSION  yes
  DATA_RECORD_COMPRESSION  yes
  DUPLICATES        no
  INDEX_AREA        1
  INDEX_COMPRESSION no
  INDEX_FILL        100
  LEVEL1_INDEX_AREA 1
  NAME              "CURRENCY_CODE"
  PROLOG            3
  SEG0_LENGTH       4
  SEG0_POSITION     0
  SEG1_LENGTH       11
  SEG1_POSITION     12
  TYPE              string

```

This is the modification that you made.

CUSTOMER.FDL

```

TITLE    "CUSTOMER"
IDENT    "29-MAY-1997 10:00:09  OpenVMS FDL Editor"
SYSTEM
SOURCE   "OpenVMS"
FILE
NAME     "CUSTOMER.DAT"
ORGANIZATION indexed
RECORD
CARRIAGE_CONTROL carriage_return
FORMAT    fixed
SIZE     379
AREA 0
  ALLOCATION          303
  BEST_TRY_CONTIGUOUS  yes
  BUCKET_SIZE       3
  EXTENSION         75

AREA 1
  ALLOCATION          3
  BEST_TRY_CONTIGUOUS  yes
  BUCKET_SIZE       3
  EXTENSION         3

KEY 0
  CHANGES           no
  DATA_AREA         0
  DATA_FILL        100
  DATA_KEY_COMPRESSION  yes
  DATA_RECORD_COMPRESSION  yes
  DUPLICATES        no
  INDEX_AREA        1
  INDEX_COMPRESSION no
  INDEX_FILL        100
  LEVEL1_INDEX_AREA 1

```

NAME	"CUSTOMER_CODE"
PROLOG	3
SEG0_LENGTH	10
SEG0_POSITION	0
TYPE	string

DESTINATION.FDL

TITLE	"DESTINATION"
IDENT	"28-MAY-1997 09:57:29 OpenVMS FDL Editor"
SYSTEM	
SOURCE	"OpenVMS"
FILE	
NAME	"DESTINATION.DAT"
ORGANIZATION	indexed
RECORD	
CARRIAGE_CONTROL	carriage_return
FORMAT	fixed
SIZE	60
AREA 0	
ALLOCATION	42
BEST_TRY_CONTIGUOUS	yes
BUCKET_SIZE	3
EXTENSION	9
AREA 1	
ALLOCATION	6
BEST_TRY_CONTIGUOUS	yes
BUCKET_SIZE	3
EXTENSION	3
KEY 0	
CHANGES	no
DATA_AREA	0
DATA_FILL	100
DATA_KEY_COMPRESSION	yes
DATA_RECORD_COMPRESSION	yes
DUPLICATES	no
INDEX_AREA	1
INDEX_COMPRESSION	no
INDEX_FILL	100
LEVEL1_INDEX_AREA	1
NAME	"DESTINATION_CODE"
PROLOG	3
SEG0_LENGTH	10
SEG0_POSITION	0
TYPE	string

INVOICE.FDL

\$ TYPE	INVOICE.FDL
IDENT	" 3-JUN-1997 11:44:22 OpenVMS ANALYZE/RMS_FILE Utility"
SYSTEM	
SOURCE	OpenVMS
FILE	
ALLOCATION	135
BEST_TRY_CONTIGUOUS	yes
BUCKET_SIZE	3
CLUSTER_SIZE	9
CONTIGUOUS	no
EXTENSION	21
FILE_MONITORING	no
GLOBAL_BUFFER_COUNT	0

```

NAME "DKA300:[TRAIN_RMS.DATA]INVOICE.DAT;2"
ORGANIZATION indexed
OWNER [CAROLYN]
PROTECTION (system:RWED, owner:RWED, group:RE, world:)

RECORD
BLOCK_SPAN yes
CARRIAGE_CONTROL carriage_return
FORMAT fixed
SIZE 101

AREA 0
ALLOCATION 81
BEST_TRY_CONTIGUOUS yes
BUCKET_SIZE 3
EXTENSION 21

AREA 1
ALLOCATION 9
BEST_TRY_CONTIGUOUS yes
BUCKET_SIZE 3
EXTENSION 3

AREA 2
ALLOCATION 45
BEST_TRY_CONTIGUOUS yes
BUCKET_SIZE 3
EXTENSION 15

KEY 0
CHANGES no
DATA_KEY_COMPRESSION yes
DATA_RECORD_COMPRESSION yes
DATA_AREA 0
DATA_FILL 100
DUPLICATES no
INDEX_AREA 1
INDEX_COMPRESSION no
INDEX_FILL 100
LEVEL1_INDEX_AREA 1
NAME "INVOICE/LINE_NUMBER"
NULL_KEY no
PROLOG 3
SEG0_LENGTH 8
SEG0_POSITION 0
TYPE string

KEY 1
CHANGES yes
DATA_KEY_COMPRESSION no
DATA_AREA 2
DATA_FILL 100
DUPLICATES yes
INDEX_AREA 2
INDEX_COMPRESSION no
INDEX_FILL 100
LEVEL1_INDEX_AREA 2
NAME "INVOICE_TYPE"
NULL_KEY no
SEG0_LENGTH 4
SEG0_POSITION 8
TYPE string

KEY 2
CHANGES yes
DATA_KEY_COMPRESSION yes
DATA_AREA 2
DATA_FILL 100
DUPLICATES yes
INDEX_AREA 2
INDEX_COMPRESSION no
INDEX_FILL 100

```

LEVEL1_INDEX_AREA	2
NAME	"INVOICE_DATE"
NULL_KEY	no
SEG0_LENGTH	11
SEG0_POSITION	12
TYPE	string
ANALYSIS_OF_AREA 0	
RECLAIMED_SPACE	0
ANALYSIS_OF_AREA 1	
RECLAIMED_SPACE	0
ANALYSIS_OF_AREA 2	
RECLAIMED_SPACE	0
ANALYSIS_OF_KEY 0	
DATA_FILL	5
DATA_KEY_COMPRESSION	18
DATA_RECORD_COMPRESSION	79
DATA_RECORD_COUNT	2
DATA_SPACE_OCCUPIED	3
DEPTH	1
INDEX_COMPRESSION	0
INDEX_FILL	1
INDEX_SPACE_OCCUPIED	3
LEVEL1_RECORD_COUNT	1
MEAN_DATA_LENGTH	101
MEAN_INDEX_LENGTH	10
ANALYSIS_OF_KEY 1	
DATA_FILL	3
DATA_KEY_COMPRESSION	0
DATA_RECORD_COUNT	2
DATA_SPACE_OCCUPIED	3
DEPTH	1
DUPLICATES_PER_SIDR	1
INDEX_COMPRESSION	0
INDEX_FILL	1
INDEX_SPACE_OCCUPIED	3
LEVEL1_RECORD_COUNT	1
MEAN_DATA_LENGTH	16
MEAN_INDEX_LENGTH	6
ANALYSIS_OF_KEY 2	
DATA_FILL	1
DATA_KEY_COMPRESSION	72
DATA_RECORD_COUNT	1
DATA_SPACE_OCCUPIED	3
DEPTH	1
DUPLICATES_PER_SIDR	1
INDEX_COMPRESSION	0
INDEX_FILL	2
INDEX_SPACE_OCCUPIED	3
LEVEL1_RECORD_COUNT	1
MEAN_DATA_LENGTH	15
MEAN_INDEX_LENGTH	13

SUPPLIER.FDL

```

$ TYPE SUPPLIER.FDL
TITLE    "SUPPLIER"
IDENT    "28-MAY-1997 09:45:20  OpenVMS FDL Editor"
SYSTEM
SOURCE      "OpenVMS"
FILE
NAME        "SUPPLIER.DAT"
ORGANIZATION indexed
RECORD
CARRIAGE_CONTROL carriage_return
FORMAT      fixed
SIZE        409
AREA 0
ALLOCATION   240
BEST_TRY_CONTIGUOUS yes
BUCKET_SIZE 6
EXTENSION   60
AREA 1
ALLOCATION   12
BEST_TRY_CONTIGUOUS yes
BUCKET_SIZE 6
EXTENSION   6
AREA 2
ALLOCATION   87
BEST_TRY_CONTIGUOUS yes
BUCKET_SIZE 3
EXTENSION   27
KEY 0
CHANGES    no
DATA_AREA   0
DATA_FILL   100
DATA_KEY_COMPRESSION yes
DATA_RECORD_COMPRESSION yes
DUPLICATES  no
INDEX_AREA  1
INDEX_COMPRESSION no
INDEX_FILE  100
LEVEL1_INDEX_AREA 1
NAME        "SUPPLIER_CODE"
PROLOG      3
SEG0_LENGTH 10
SEG0_POSITION 0
TYPE        string
KEY 1
CHANGES    yes
DATA_AREA   2
DATA_FILL   100
DATA_KEY_COMPRESSION yes
DUPLICATES  yes
INDEX_AREA  2
INDEX_COMPRESSION no
INDEX_FILE  100
LEVEL1_INDEX_AREA 2
NAME        "NAME"
SEG0_LENGTH 40
SEG0_POSITION 10
TYPE        string
KEY 2
CHANGES    yes

```



```

DATA_AREA          2
DATA_FILL          100
DATA_KEY_COMPRESSION  yes
DUPLICATES         yes
INDEX_AREA         2
INDEX_COMPRESSION  no
INDEX_FILL         100
LEVEL1_INDEX_AREA  2
NAME               "TOWN"
SEG0_LENGTH        30
SEG0_POSITION      200
TYPE               string

KEY 3
CHANGES           yes
DATA_AREA          2
DATA_FILL          100
DATA_KEY_COMPRESSION  yes
DUPLICATES         yes
INDEX_AREA         2
INDEX_COMPRESSION  no
INDEX_FILL         100
LEVEL1_INDEX_AREA  2
NAME               "POSTCODE"
SEG0_LENGTH        20
SEG0_POSITION      260
TYPE               string

```

WORLD_DESTINATION.FDL

```

$ TYPE WORLD_DESTINATION.FDL
TITLE   "world_destination"
IDENT   "21-AUG-1997 16:38:09  OpenVMS FDL Editor"
SYSTEM
SOURCE   "OpenVMS"

FILE
NAME
"easysoft_sql_data:world_destination.dat"
ORGANIZATION  indexed

RECORD
CARRIAGE_CONTROL  carriage_return
FORMAT            fixed
SIZE              43

AREA 0
ALLOCATION          42
BEST_TRY_CONTIGUOUS  yes
BUCKET_SIZE        3
EXTENSION          9

AREA 1
ALLOCATION          6
BEST_TRY_CONTIGUOUS  yes
BUCKET_SIZE        3
EXTENSION          3

AREA 2
ALLOCATION          24
BEST_TRY_CONTIGUOUS  yes
BUCKET_SIZE        3
EXTENSION          9

KEY 0
CHANGES           no
DATA_AREA          0
DATA_FILL          100
DATA_KEY_COMPRESSION  yes

```

```
DATA_RECORD_COMPRESSION yes
DUPLICATES                no
INDEX_AREA                1
INDEX_COMPRESSION         no
INDEX_FILL                100
LEVEL1_INDEX_AREA        1
NAME                      "world_Destination"
PROLOG                    3
SEG0_LENGTH               13
SEG0_POSITION             0
TYPE                      string

KEY 1
CHANGES                  no
DATA_AREA                 2
DATA_FILL                 100
DATA_KEY_COMPRESSION      yes
DUPLICATES                yes
INDEX_AREA                2
INDEX_COMPRESSION         no
INDEX_FILL                100
LEVEL1_INDEX_AREA        2
NAME                      "destination"
SEG0_LENGTH               10
SEG0_POSITION             3
TYPE                      string
```

CSV Description

This section shows the CSV for the objects (e.g. files, fields, users) in the catalog. For ease of reading, this has been split into sections for each of the objects and the field names are shown at the start of each section. The sections are presented in the order that the CSV definitions appear.

File Definitions

The file definitions consist of the following fields:

"FILE", "<File Name>", "<File Organisation>", "<Record Type>", "<Record Size>"

```
"FILE", "BROCHURE", "INDEXED", "FIXED", 275
"FILE", "CURRENCY_MASTER", "INDEXED", "FIXED", 58
"FILE", "CURRENCY_RATE", "INDEXED", "FIXED", 23
"FILE", "CUSTOMER", "INDEXED", "FIXED", 379
"FILE", "DESTINATION", "INDEXED", "FIXED", 60
"FILE", "DETAILS", "INDEXED", "FIXED", 101
"FILE", "HEADER", "INDEXED", "FIXED", 101
"FILE", "SUPPLIER", "INDEXED", "FIXED", 409
"FILE", "WORLD_DESTINATION", "INDEXED", "FIXED", 43 "FILE", "BROCHURE", "INDEXED", "FIXED", 275
```

Field Definitions

The field definitions consist of the following fields:

"FIELD", "<File Name>", "<Field Name>", "<Data Type>", "<Offset>", "<Length>", "<Precision>", "<Scale>", "<Encrypted>", "<Date Format>", "<Default>"

Note: <Date Format> is only used for ASCII Dates/Times.

```
"FIELD", "BROCHURE", "BROCHURE_CODE", "STRING", 0, 10, 10, 0, 0, "", ""
"FIELD", "BROCHURE", "SUPPLIER_CODE", "STRING", 10, 10, 10, 0, 0, "", ""
"FIELD", "BROCHURE", "DESCRIPTION", "STRING", 20, 254, 254, 0, 0, "", ""
"FIELD", "CURRENCY_MASTER", "CURRENCY_CODE", "STRING", 0, 4, 8, 0, 0, "", ""
"FIELD", "CURRENCY_MASTER", "TITLE", "STRING", 4, 30, 30, 0, 0, "", ""
"FIELD", "CURRENCY_MASTER", "UNITS_PLURAL", "STRING", 34, 12, 12, 0, 0, "", ""
"FIELD", "CURRENCY_MASTER", "UNITS_SINGULAR", "STRING", 46, 12, 12, 0, 0, "", ""
"FIELD", "CURRENCY_RATE", "CURRENCY_CODE", "STRING", 0, 4, 4, 0, 0, "", ""
"FIELD", "CURRENCY_RATE", "CURRENCY_RATE", "DOUBLE", 4, 8, 15, 0, 0, "", ""
"FIELD", "CURRENCY_RATE", "CURRENCY_DATE", "DATE", 12, 11, 11, 0, 0, "DD-MMM-YYYY", "01-JAN-1997"
"FIELD", "CUSTOMER", "CUSTOMER_CODE", "STRING", 0, 10, 10, 0, 0, "", ""
"FIELD", "CUSTOMER", "NAME", "STRING", 10, 30, 30, 0, 0, "", ""
"FIELD", "CUSTOMER", "FORENAME", "STRING", 40, 30, 30, 0, 0, "", ""
"FIELD", "CUSTOMER", "TITLE", "STRING", 70, 4, 4, 0, 0, "", ""
"FIELD", "CUSTOMER", "ADDRESS1", "STRING", 74, 50, 50, 0, 0, "", ""
"FIELD", "CUSTOMER", "ADDRESS2", "STRING", 124, 50, 50, 0, 0, "", ""
"FIELD", "CUSTOMER", "ADDRESS3", "STRING", 174, 50, 50, 0, 0, "", ""
"FIELD", "CUSTOMER", "TOWN", "STRING", 224, 30, 30, 0, 0, "", ""
"FIELD", "CUSTOMER", "COUNTY", "STRING", 254, 30, 30, 0, 0, "", ""
"FIELD", "CUSTOMER", "POSTCODE", "STRING", 284, 15, 15, 0, 0, "", ""
"FIELD", "CUSTOMER", "TELEPHONE", "STRING", 299, 15, 15, 0, 0, "", ""
"FIELD", "CUSTOMER", "FAX", "STRING", 314, 15, 15, 0, 0, "", ""
"FIELD", "CUSTOMER", "COMMENTS", "STRING", 329, 50, 50, 0, 0, "", ""
"FIELD", "DESTINATION", "DESTINATION_CODE", "STRING", 0, 10, 10, 0, 0, "", ""
"FIELD", "DESTINATION", "DESCRIPTION", "STRING", 10, 50, 50, 0, 0, "", ""
"FIELD", "DETAILS", "INVOICE_NUMBER", "INTEGER4", 0, 4, 10, 0, 0, "", ""
"FIELD", "DETAILS", "LINE_NUMBER", "INTEGER4", 4, 4, 10, 0, 0, "", ""
```

```

"FIELD", "DETAILS", "FILLER", "STRING", 8, 15, 15, 0, 0, "", ""
"FIELD", "DETAILS", "BROCHURE_CODE", "STRING", 23, 10, 10, 0, 0, "", ""
"FIELD", "DETAILS", "DESTINATION_CODE", "STRING", 33, 10, 10, 0, 0, "", ""
"FIELD", "DETAILS", "DESCRIPTION", "STRING", 43, 50, 50, 0, 0, "", ""
"FIELD", "DETAILS", "VALUE", "DOUBLE", 93, 8, 15, 0, 0, "", ""
"FIELD", "HEADER", "INVOICE_NUMBER", "INTEGER4", 0, 4, 10, 0, 0, "", ""
"FIELD", "HEADER", "LINE_NUMBER", "INTEGER4", 4, 4, 10, 0, 0, "", ""
"FIELD", "HEADER", "INVOICE_TYPE", "STRING", 8, 4, 4, 0, 0, "", ""
"FIELD", "HEADER", "INVOICE_DATE", "DATE", 12, 11, 11, 0, 0, "DD-MMM-YYYY", ""
"FIELD", "HEADER", "INVOICE_VALUE", "DOUBLE", 23, 8, 15, 0, 0, "", ""
"FIELD", "HEADER", "CODE", "STRING", 31, 10, 10, 0, 0, "", ""
"FIELD", "HEADER", "CURRENCY", "STRING", 41, 4, 4, 0, 0, "", ""
"FIELD", "SUPPLIER", "SUPPLIER_CODE", "STRING", 0, 10, 10, 0, 0, "", ""
"FIELD", "SUPPLIER", "NAME", "STRING", 10, 40, 40, 0, 0, "", ""
"FIELD", "SUPPLIER", "ADDRESS1", "STRING", 50, 50, 50, 0, 0, "", ""
"FIELD", "SUPPLIER", "ADDRESS2", "STRING", 100, 50, 50, 0, 0, "", ""
"FIELD", "SUPPLIER", "ADDRESS3", "STRING", 150, 50, 50, 0, 0, "", ""
"FIELD", "SUPPLIER", "TOWN", "STRING", 200, 30, 30, 0, 0, "", ""
"FIELD", "SUPPLIER", "COUNTY", "STRING", 230, 30, 30, 0, 0, "", ""
"FIELD", "SUPPLIER", "POSTCODE", "STRING", 260, 20, 20, 0, 0, "", ""
"FIELD", "SUPPLIER", "PHONE", "STRING", 280, 15, 15, 0, 0, "", ""
"FIELD", "SUPPLIER", "FAX", "STRING", 295, 15, 15, 0, 0, "", ""
"FIELD", "SUPPLIER", "COMMENTS", "STRING", 310, 15, 15, 0, 0, "", ""
"FIELD", "SUPPLIER", "CONTACT_NAME", "STRING", 325, 30, 30, 0, 0, "", ""
"FIELD", "SUPPLIER", "CONTACT_TITLE", "STRING", 355, 4, 4, 0, 0, "", ""
"FIELD", "SUPPLIER", "CONTACT_FORENAME", "STRING", 359, 30, 30, 0, 0, "", ""
"FIELD", "SUPPLIER", "CONTACT_PHONE", "STRING", 389, 15, 15, 0, 0, "", ""
"FIELD", "SUPPLIER", "CONTACT_EXTENSION", "STRING", 404, 5, 5, 0, 0, "", ""
"FIELD", "WORLD_DESTINATION", "WORLD_CODE", "STRING", 0, 3, 3, 0, 0, "", ""
"FIELD", "WORLD_DESTINATION", "DESTINATION_CODE", "STRING", 3, 10, 10, 0, 0, "", ""
"FIELD", "WORLD_DESTINATION", "TITLE", "STRING", 13, 30, 30, 0, 0, "", ""

```

Database Definitions

The database definitions consist of the following fields:

```
"DB", "<Database Name>", "<Default Directory>", "<Driver Name>", "<Connect String>"
```

```
"DB", "TRAINING", "EASYSOFT_SQL_DATA", "VAX-RMS", ""
```

Table Definitions

The table definitions consist of the following fields:

```
"TABLE", "<Database Name>", "<Table Name>", "<Table Type>", "<File Name>",
"<File Specification>", "<Criteria>"
```

```

"TABLE", "TRAINING", "BROCHURE", "TABLE", "BROCHURE", "BROCHURE.DAT", ""
"TABLE", "TRAINING", "CURRENCY_MASTER", "TABLE", "CURRENCY_MASTER", "CURRENCY_MAST
ER.DAT", ""
"TABLE", "TRAINING", "CURRENCY_RATE", "TABLE", "CURRENCY_RATE", "CURRENCY_RATE.DAT
", ""
"TABLE", "TRAINING", "CUSTOMER", "TABLE", "CUSTOMER", "CUSTOMER.DAT", ""
"TABLE", "TRAINING", "DESTINATION", "TABLE", "DESTINATION", "DESTINATION.DAT", ""
"TABLE", "TRAINING", "DETAILS", "TABLE", "DETAILS", "INVOICE.DAT", "TRAINING_DETAIL
S.LINE_NUMBER>0"
"TABLE", "TRAINING", "HEADER", "TABLE", "HEADER", "INVOICE.DAT", "TRAINING_HEADER.L
INE_NUMBER=0"
"TABLE", "TRAINING", "SUPPLIER", "TABLE", "SUPPLIER", "SUPPLIER.DAT", ""
"TABLE", "TRAINING", "WORLD_DESTINATION", "TABLE", "WORLD_DESTINATION", "WORLD_DES
TINATION.DAT", ""

```

Column Definitions

The column definitions consist of the following fields:

"COLUMN" , "<Database Name>" , "<Table Name>" , "<Column Name>" , "<Field Name>" ,
 "<SQL Data Type>" , <Length> , <Updatable> , <Visible> , "<Default Value>"

```
"COLUMN" , "TRAINING" , "BROCHURE" , "BROCHURE_CODE" , "BROCHURE_CODE" , "VARCHAR" , 10 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "BROCHURE" , "SUPPLIER_CODE" , "SUPPLIER_CODE" , "VARCHAR" , 10 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "BROCHURE" , "DESCRIPTION" , "DESCRIPTION" , "VARCHAR" , 254 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "CURRENCY_MASTER" , "CURRENCY_CODE" , "CURRENCY_CODE" , "VARCHAR" , 8 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "CURRENCY_MASTER" , "TITLE" , "TITLE" , "VARCHAR" , 30 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "CURRENCY_MASTER" , "UNITS_PLURAL" , "UNITS_PLURAL" , "VARCHAR" , 12 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "CURRENCY_MASTER" , "UNITS_SINGULAR" , "UNITS_SINGULAR" , "VARCHAR" , 12 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "CURRENCY_RATE" , "CURRENCY_CODE" , "CURRENCY_CODE" , "VARCHAR" , 4 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "CURRENCY_RATE" , "CURRENCY_RATE" , "CURRENCY_RATE" , "DOUBLE" , 8 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "CURRENCY_RATE" , "CURRENCY_DATE" , "CURRENCY_DATE" , "DATE" , 6 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "CUSTOMER" , "CUSTOMER_CODE" , "CUSTOMER_CODE" , "VARCHAR" , 10 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "CUSTOMER" , "NAME" , "NAME" , "VARCHAR" , 30 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "CUSTOMER" , "FORENAME" , "FORENAME" , "VARCHAR" , 30 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "CUSTOMER" , "TITLE" , "TITLE" , "VARCHAR" , 4 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "CUSTOMER" , "ADDRESS1" , "ADDRESS1" , "VARCHAR" , 50 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "CUSTOMER" , "ADDRESS2" , "ADDRESS2" , "VARCHAR" , 50 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "CUSTOMER" , "ADDRESS3" , "ADDRESS3" , "VARCHAR" , 50 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "CUSTOMER" , "TOWN" , "TOWN" , "VARCHAR" , 30 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "CUSTOMER" , "COUNTY" , "COUNTY" , "VARCHAR" , 30 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "CUSTOMER" , "POSTCODE" , "POSTCODE" , "VARCHAR" , 15 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "CUSTOMER" , "TELEPHONE" , "TELEPHONE" , "VARCHAR" , 15 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "CUSTOMER" , "FAX" , "FAX" , "VARCHAR" , 15 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "CUSTOMER" , "COMMENTS" , "COMMENTS" , "VARCHAR" , 50 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "DESTINATION" , "DESTINATION_CODE" , "DESTINATION_CODE" , "VARCHAR" , 10 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "DESTINATION" , "DESCRIPTION" , "DESCRIPTION" , "VARCHAR" , 50 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "DETAILS" , "INVOICE_NUMBER" , "INVOICE_NUMBER" , "INTEGER" , 4 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "DETAILS" , "LINE_NUMBER" , "LINE_NUMBER" , "INTEGER" , 4 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "DETAILS" , "BROCHURE_CODE" , "BROCHURE_CODE" , "VARCHAR" , 10 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "DETAILS" , "DESTINATION_CODE" , "DESTINATION_CODE" , "VARCHAR" , 10 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "DETAILS" , "DESCRIPTION" , "DESCRIPTION" , "VARCHAR" , 50 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "DETAILS" , "VALUE" , "VALUE" , "DOUBLE" , 8 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "HEADER" , "INVOICE_NUMBER" , "INVOICE_NUMBER" , "INTEGER" , 4 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "HEADER" , "LINE_NUMBER" , "LINE_NUMBER" , "INTEGER" , 4 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "HEADER" , "INVOICE_TYPE" , "INVOICE_TYPE" , "VARCHAR" , 4 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "HEADER" , "INVOICE_DATE" , "INVOICE_DATE" , "DATE" , 6 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "HEADER" , "INVOICE_VALUE" , "INVOICE_VALUE" , "DOUBLE" , 8 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "HEADER" , "CODE" , "CODE" , "VARCHAR" , 10 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "HEADER" , "CURRENCY" , "CURRENCY" , "VARCHAR" , 4 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "SUPPLIER" , "SUPPLIER_CODE" , "SUPPLIER_CODE" , "VARCHAR" , 10 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "SUPPLIER" , "NAME" , "NAME" , "VARCHAR" , 40 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "SUPPLIER" , "ADDRESS1" , "ADDRESS1" , "VARCHAR" , 50 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "SUPPLIER" , "ADDRESS2" , "ADDRESS2" , "VARCHAR" , 50 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "SUPPLIER" , "ADDRESS3" , "ADDRESS3" , "VARCHAR" , 50 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "SUPPLIER" , "TOWN" , "TOWN" , "VARCHAR" , 30 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "SUPPLIER" , "COUNTY" , "COUNTY" , "VARCHAR" , 30 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "SUPPLIER" , "POSTCODE" , "POSTCODE" , "VARCHAR" , 20 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "SUPPLIER" , "PHONE" , "PHONE" , "VARCHAR" , 15 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "SUPPLIER" , "FAX" , "FAX" , "VARCHAR" , 15 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "SUPPLIER" , "COMMENTS" , "COMMENTS" , "VARCHAR" , 15 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "SUPPLIER" , "CONTACT_NAME" , "CONTACT_NAME" , "VARCHAR" , 30 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "SUPPLIER" , "CONTACT_TITLE" , "CONTACT_TITLE" , "VARCHAR" , 4 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "SUPPLIER" , "CONTACT_FORENAME" , "CONTACT_FORENAME" , "VARCHAR" , 30 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "SUPPLIER" , "CONTACT_PHONE" , "CONTACT_PHONE" , "VARCHAR" , 15 , 1 , 1 , ""
```

```
"COLUMN" , "TRAINING" , "SUPPLIER" , "CONTACT_EXTENSION" , "CONTACT_EXTENSION" , "VARCHAR" , 5 , 1
, 1 , ""
"COLUMN" , "TRAINING" , "WORLD_DESTINATION" , "WORLD_CODE" , "WORLD_CODE" , "VARCHAR" , 3 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "WORLD_DESTINATION" , "DESTINATION_CODE" , "DESTINATION_CODE" , "VARCH
AR" , 10 , 1 , 1 , ""
"COLUMN" , "TRAINING" , "WORLD_DESTINATION" , "TITLE" , "TITLE" , "VARCHAR" , 30 , 1 , 1 , ""
```

User Definitions

The user definitions consist of the following fields:

```
"USER" , "<User Name>" , "<Password>"
```

Note: The password field is not written on exports.

```
"USER" , "TRAINEEn" , ""
```

Database Privilege Definitions

The database privilege definitions consist of the following fields:

```
"DBPRV" , "<User Name>" , "<Database Name>" , "<Database Username>" , "<Database Password>" ,
<Allow Select> , <Allow Insert> , <Allow Update> , <Allow Delete> , "<Security Level>"
```

```
"DBPRV" , "ADMIN" , "TRAINING" , "" , "" , 1 , 1 , 1 , 1 , "DATABASE"
"DBPRV" , "TRAINEE1" , "TRAINING" , "" , "" , 1 , 1 , 1 , 1 , "DATABASE"
```

Table Privileges Definitions

Table Privilege definitions consist of the following fields:

```
"TBLPRV" , "<Database Username>" , "<Database Name>" , "<Database Password>" , "<Table Name>"
, <Allow Select> , <Allow Insert> , <Allow Update> , <Allow Delete>
```

There are no definitions for the TRAINEEn user.

11. RMS Exercises

In this module you will use Easysoft ODBC for RMS in conjunction with ODBC-compliant applications to retrieve and upload data to RMS files on the server.

Contents

General Information	11-2
Tips and Reminders	11-2
Task 1 : Produce a list of all Suppliers	11-4
Task 2 : Produce a List of Supplier Names	11-5
Task 3 : Produce a directory of all Suppliers	11-6
Task 4 : Produce a directory of Virgin Holidays Ltd (S0012)	11-7
Task 5 : Produce a list of addresses of the Suppliers in London	11-9
Task 6 : List all Spanish Destinations	11-10
Task 7 : List CUSTOMERS with the name Smith or Jones	11-11
Task 8 : List the Brochures and the Suppliers they belong to	11-13
Task 9 : List all brochures for Thomson Holidays Ltd	11-15
Task 10 : List existing currencies and currency rates	11-17
Task 11 : Find the currency used in Cancun	11-21
Task 12 : How many customers are there?	11-23
Task 13 : Update the CUSTOMER file	11-24
Task 14 : Insert a New Supplier Record	11-25
Task 15 : Delete a Supplier record	11-26
Task 16 : Find the Holiday company we have spent the most money with	11-27
Task 17 : Find the most popular holiday destination	11-30
Task 18 : Prepare a report to show the Total Profits	11-32
Task 19 : Prepare a letter to all Clients who spent over £5000.00 in May	11-34
Task 20 : Produce a Purchase Invoice form	11-36
Task 21 : Insert data into an RMS file from local Access Tables	11-42

General Information

The purpose of the exercises (also called tasks) is to show you an easy way to retrieve and update RMS data. We do not concern ourselves with the manipulation and formatting of data. In these exercises, you will use a combination of Microsoft Excel, Microsoft Access and Microsoft Word Mail Merge to connect to the "Training Data" data source which you previously set up. Unless otherwise stated, the steps listed are based on Microsoft Excel, but they are easily applicable to Microsoft Access.




All of the exercises follow the same format; they have been designed to introduce you quickly to the concepts of using Easysoft ODBC for RMS. Each task contains a series of steps followed by the results expected. In cases where many rows are returned, then only a fragment of the result is shown. As you progress further you should try to use the steps less and less. Also, where appropriate, there are additional exercises which can be used to take the tasks further.

Tips and Reminders

The best approach to solving the exercises is KISS (Keep it Simple, Sweetie).



The table on the next page gives you a few reminders and hints for Excel/Query and Access.

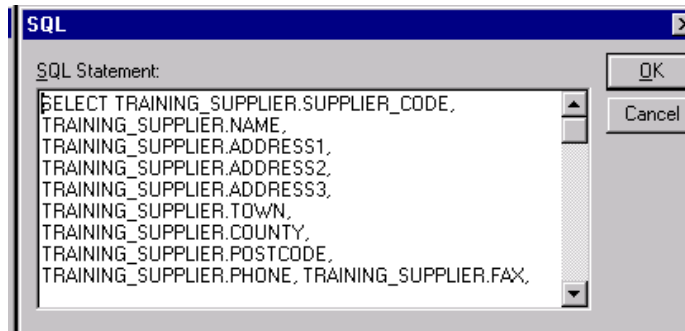
	 Microsoft Excel/Query	 Microsoft Access
Wildcard in queries	% represents zero or more characters. _ represents any single alphabetic character.	* represents zero or more characters. ? represents any single alphabetic character.
Change column heading	To make a report more comprehensible, you can change the names of columns that are returned to the spreadsheet when they are not clear. To do this, double click on the column heading within Query and type the new column name.	In Query design mode, prefix the field name with the name that is to appear as the column heading. Separate the two names with a colon i.e. <column name>:<field name>
Allow update of records	You cannot update data directly using Excel (it is possible with the use of a macro). However, since Excel uses Microsoft Query to access the data, you can upload data from Query, provided you have not returned to Excel. To allow updates, select Records, Allow Editing in Query.	Use the Link Tables... option (rather than Import). A connection is made to the data files, and the data can be updated directly. In Query mode, use Query, Update
Record update/delete/insert	If a record is inserted, deleted or updated, then Query re-reads the entire file, which can be a slow process.	If a record is inserted, deleted or updated, then Access does not re-read the entire file.
Selecting columns	In Microsoft Query, to add a column to the Data Pane from a table which is in the Table Pane, you can either highlight the column and drag it into the Data Pane, or you can double-click on the column.	To add a column to the Design Grid from a table, you can either highlight the column and drag it into the Data Pane, or you can double-click on the column.
Connect to data source	Each time that you return data to Excel from Query or each time you change a query, Excel reconnects to the data source.	-
Optimisation	Query automatically executes the tables with in the FROM clause in alphabetical order, this can affect the optimisation of your query.	Access executes the tables with in the FROM clause in the order in which they are added to the query. This can affect the optimisation of your query. In Query mode, use Query, SQL Specific, Pass-Through to execute SQL directly.
View SQL	View, SQL or click the  button.	In query mode: View, SQL
View indexes	Not applicable.	From the table: View, Table Design, then View, Indexes From the database: select table, click Design, then View, Indexes
Other notes	-	To use the COUNT(*) function the Show Totals option needs to be selected.

Task 1 : Produce a list of all Suppliers

Step One	Add the SUPPLIER table to the query
Step Two	To select all the columns in the table, highlight the asterisk (*) and drag it into the Data Pane (or double-click on the column)

Result Set

SUPPLIER_CODE	NAME	ADDRESS1	ADDRESS2	ADDRESS3	TOWN	COUNTY
S0001	Thomson Holidays Ltd	Greater London House	Hamstead Road		London	Greater London
S0002	Airtous Holidays Ltd	Wavell House	Holcombe Road		Helmshore	Lancashire
S0003	Cosmosair Plc	Tourama House	17 Homesdale Road		Bromley	Kent
S0004	Kuoni Travel	5 Exchange Street			Manchester	Greater Manchester
S0005	Cresta Holidays	Tabley Court	Victoria Street		Altonham	Cheshire
S0006	Majic of Italy	227 Shepherds Bush Rd			London	Greater London
S0007	Best Western	Vine House	143 London Road		Kingston Upon Thames	Surrey
S0008	Trek America	4 Waterperry Court	Middleton Road	Banbury	Oxon	Oxfordshire
S0009	North American Travel Services	Suite 2000	11 Albion Street		Leeds	West Yorkshire
S0010	American Airlines Holidays	Trinity Square	23-9 Staines Road		Hunslow	Middlesex
S0011	Bales Tours Ltd	Bales House	Junction Road		Dorking	Surrey
S0012	Virgin Holidays Ltd	The Galleria	Ground Floor	Station Road	Crawley	West Sussex
S0013	P&O Holidays	77 New Oxford Street			London	Greater London



Notice how Excel has listed all of the columns.

The SQL could also be written as follows:

```
SELECT * FROM TRAINING_SUPPLIER
```

Task 2 : Produce a List of Supplier Names

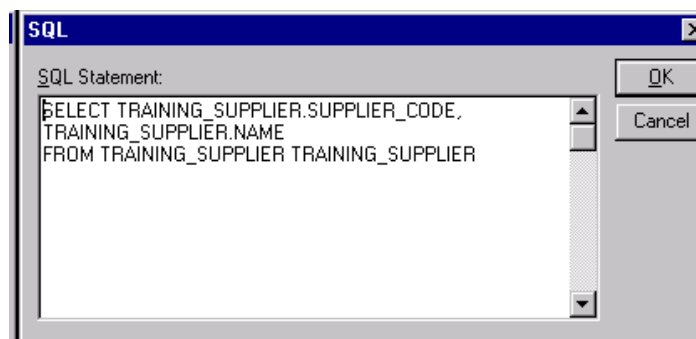
Note: The result should contain the columns , SUPPLIER_CODE, NAME.

Step One	Add the SUPPLIER table to the query
Step Two	Turn off Auto Query
Step Three	Add the SUPPLIER_CODE column to the query
Step Four	Add the NAME column to the query
Step Six	Click Run Query

Result Set

SUPPLIER_CODE	NAME	
S0001	Thomson Holidays Ltd	
S0002	Airtous Holidays Ltd	
S0003	Cosmosair Plc	
S0004	Kuoni Travel	
S0005	Cresta Holidays	
S0006	Majic of Italy	
S0007	Best Western	
S0008	Trek America	
S0009	North American Travel Service	
S0010	American Airlines Holidays Ltd	
S0011	Bales Tours Ltd	
S0012	Virgin Holidays Ltd	
S0013	P&O Holidays	

SQL



Task 3 : Produce a directory of all Suppliers

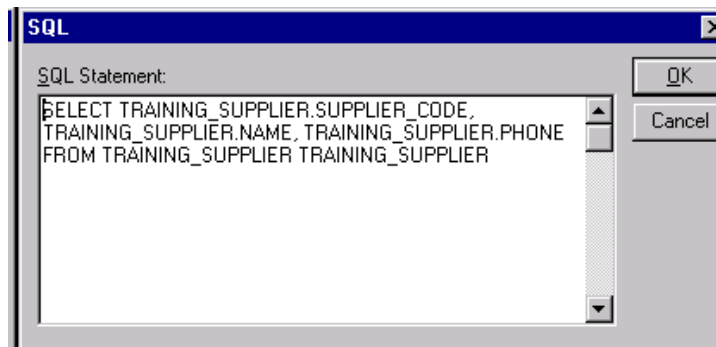
Note: A directory includes the supplier code, name and telephone number.

Step One	Add the SUPPLIER table to the query
Step Two	Turn off Auto Query
Step Three	Add the SUPPLIER_CODE column to the query
Step Four	Add the NAME and PHONE columns into the query
Step Five	Click Run Query

Result Set

	SUPPLIER_CODE	NAME	PHONE
▶	S0001	Thomson Holidays Ltd	0990 502399
	S0002	Airtous Holidays Ltd	01706 420000
	S0003	Cosmosair Plc	0181 3131941
	S0004	Kuoni Travel	0161 8320667
	S0005	Cresta Holidays	0161 9290000
	S0006	Majic of Italy	0181 748 7575
	S0007	Best Western	0181 5471515
	S0008	Trek America	
	S0009	North American Travel Service	0113 2461466
	S0010	American Airlines Holidays Ltd	0181 5779966
	S0011	Bales Tours Ltd	01306 885991
	S0012	Virgin Holidays Ltd	01293 617181
	S0013	P&O Holidays	0171 8002222

SQL



Task 4 : Produce a directory of Virgin Holidays Ltd (S0012)

Note: A directory includes the Supplier code, Name and Telephone Number.

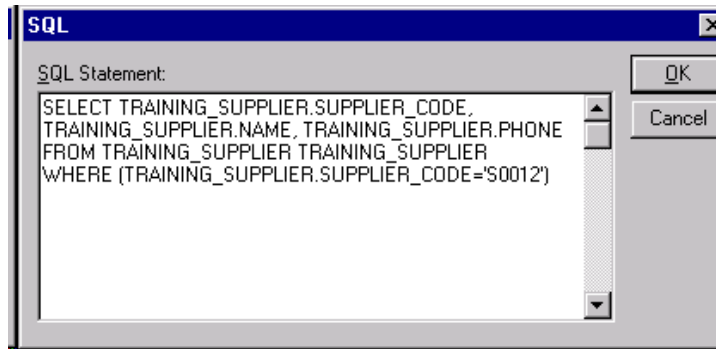
Step One	Add the SUPPLIER table to the query
Step Two	Turn off Auto Query
Step Three	Add the SUPPLIER_CODE column to the query
Step Four	Add the NAME and PHONE columns to the query
Step Five	Use the Add Criteria dialog box to select data where SUPPLIER_CODE equals S0012 (see below). Click Add to add the criteria to the query



Step Six	Click Run Query
----------	------------------------

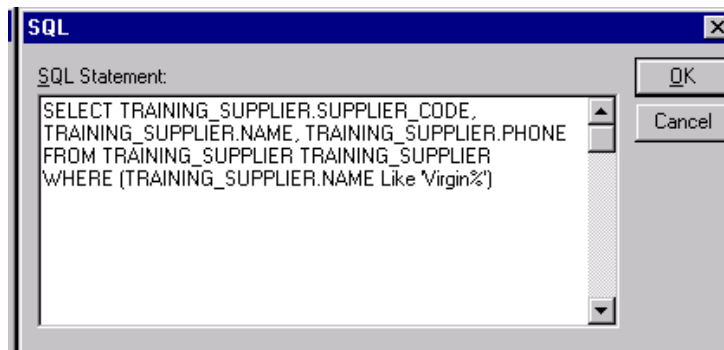
Result Set

	SUPPLIER_CODE	NAME	PHONE	
	S0012	Virgin Holidays Ltd	01293 617181	

SQL

SUGGESTED EXERCISE

Use the NAME column to add criteria instead of CODE. Use like with a wildcard e.g. Like Virgin%, to save typing the whole name. This is useful if the SUPPLIER_CODE is unknown (SQL is shown below).



Task 5 : Produce a list of addresses of the Suppliers in London

Note: Use POST_CODE to determine this. All Greater London Postcodes begin with WC. (TOWN could also be used as it is part of the key).

Step One	Add the SUPPLIER table to the query
Step Two	Turn off Auto Query
Step Three	Add the SUPPLIER_CODE column to the query
Step Four	Add the NAME, ADDRESS1, ADDRESS2, ADDRESS3, TOWN and PHONE columns to the query
Step Five	Add a criterion to select only data where POST_CODE begins with WC. Use LIKE with a wildcard.



Criteria Field:	POSTCODE
Value:	Like 'WC%'
or:	

Step Six	Click Run Query
----------	------------------------

Result Set

SUPPLIER_CODE	NAME	ADDRESS1	ADDRESS2	ADDRESS3	TOWN	PHONE
S0013	P&O Holidays	77 New Oxford Street			London	0171
S0001	Thomson Holidays Lb	Greater London House	Hamstead Road		London	099C
S0006	Majic of Italy	227 Shepherds Bush Road			London	0181



SQL	
SQL Statement:	OK Cancel
<pre> SELECT TRAINING_SUPPLIER.SUPPLIER_CODE, TRAINING_SUPPLIER.NAME, TRAINING_SUPPLIER.ADDRESS1, TRAINING_SUPPLIER.ADDRESS2, TRAINING_SUPPLIER.ADDRESS3, TRAINING_SUPPLIER.TOWN, TRAINING_SUPPLIER.PHONE FROM TRAINING_SUPPLIER TRAINING_SUPPLIER WHERE (TRAINING_SUPPLIER.POSTCODE Like 'WC%') </pre>	

Task 6 : List all Spanish Destinations

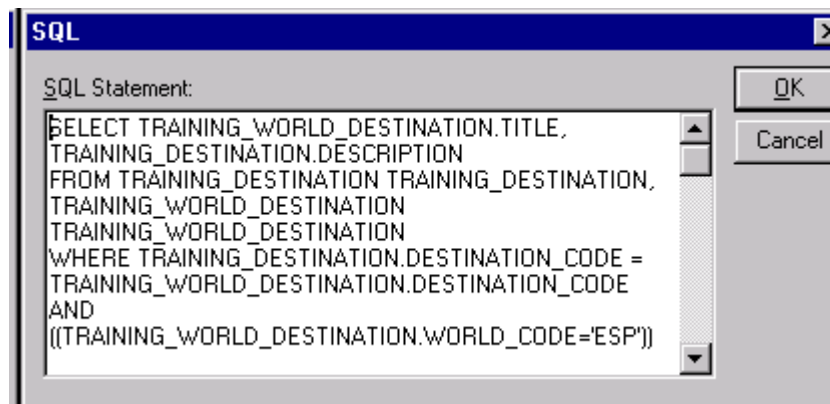
Note: Spanish World Code is ESP. Include the description of the destination.

Step One	Add the tables WORLD_DESTINATION and DESTINATION to the query
Step Two	Turn off Auto Query
Step Three	Ensure the tables are joined correctly (i.e. on DESTINATION_CODE)
Step Four	Add criteria to select Spain (WORLD_CODE = ESP)
Step Five	Add the columns TITLE and DESCRIPTION to the query
Step Six	Run the query

Result Set

TITLE	DESCRIPTION
Spain	Benidorm
Spain	Barcelona
Spain	Benalmedina
Spain	Torremolinos
Spain	Andorra
Spain	Magaluf
Spain	Playa De Las Americas
Spain	Las Palmas
Spain	Gibraltar
Spain	Granada
Spain	Madeira
Spain	Madrid
Spain	Salamanca
Spain	Seville
Spain	Nerja

SQL



The screenshot shows a window titled "SQL" with a text area containing the following SQL statement:

```
SELECT TRAINING_WORLD_DESTINATION.TITLE,
TRAINING_DESTINATION.DESCRPTION
FROM TRAINING_DESTINATION TRAINING_DESTINATION,
TRAINING_WORLD_DESTINATION
TRAINING_WORLD_DESTINATION
WHERE TRAINING_DESTINATION.DESTINATION_CODE =
TRAINING_WORLD_DESTINATION.DESTINATION_CODE
AND
((TRAINING_WORLD_DESTINATION.WORLD_CODE='ESP'))
```

Buttons for "OK" and "Cancel" are visible on the right side of the window.

Task 7 : List CUSTOMERS with the name Smith or Jones

Note: In the listing show surname, forename and title.

Step One	Add the CUSTOMER table to the query
Step Two	Turn off Auto Query
Step Three	Add the NAME column to the query
Step Four	Add the FORENAME column to the query
Step Five	Add the TITLE column to the query
Step Six	Add a criterion to select only data where NAME equals Smith

The screenshot shows the 'Add Criteria' dialog box with the following settings:

- Logic: And Or
- Total: [Empty dropdown]
- Field: NAME
- Operator: equals
- Value: Smith
- Buttons: Add, Close, Values...

Step Seven	Click Add then select OR and Add criterion to select only data where NAME equals Jones
------------	---

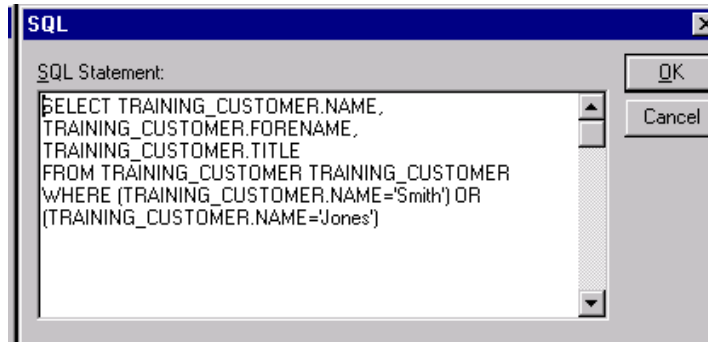
The screenshot shows the 'Add Criteria' dialog box with the following settings:

- Logic: And Or
- Total: [Empty dropdown]
- Field: NAME
- Operator: equals
- Value: Jones
- Buttons: Add, Close, Values...

Step Eight	Run the query
------------	---------------

Result Set

	NAME	FORENAME	TITLE	
	Jones	Alison	Miss	
	Smith	Guy	Mr	
	Smith	Martin	Mr	

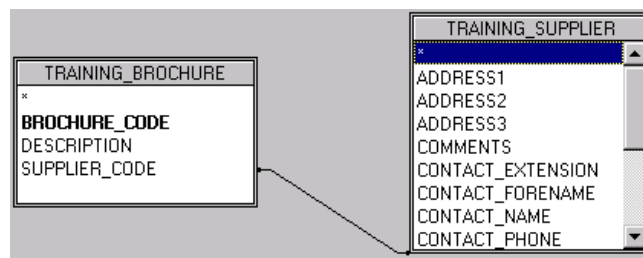
SQL

```
SQL Statement:
SELECT TRAINING_CUSTOMER.NAME,
TRAINING_CUSTOMER.FORENAME,
TRAINING_CUSTOMER.TITLE
FROM TRAINING_CUSTOMER TRAINING_CUSTOMER
WHERE (TRAINING_CUSTOMER.NAME='Smith') OR
(TRAINING_CUSTOMER.NAME='Jones')
```

Notice how parentheses (brackets) have been inserted around the two criteria. This shows how the operators are grouped (see “Precedence Order” in the module “Introduction to SQL”), and also makes for easier reading of the SQL. In this case, even without parentheses, there is no possibility of ambiguity, but where there are many levels of AND and OR, it becomes difficult to understand the logic unless they are used.

Task 8 : List the Brochures and the Suppliers they belong to

Step One	Add the BROCHURE table to the query
Step Two	Turn off Auto Query
Step Three	Add the BROCHURE_CODE column to the query
Step Four	Add the DESCRIPTION column to the query
Step Five	Add the SUPPLIER table to the query
Step Six	Ensure that the tables join on the correct fields (SUPPLIER_CODE in this case)



Step Seven	Double click on NAME to add the column to the query.
Step Eight	Run the query

Result Set

BROCHURE_CODE	DESCRIPTION	NAME	
B0001	Faraway Shores	Thomson Holidays Ltd	
B0002	Ski-ing	Thomson Holidays Ltd	
B0003	Young At Heart	Thomson Holidays Ltd	
B0004	Price Breakers	Thomson Holidays Ltd	
B0005	City Breaks	Thomson Holidays Ltd	
B0006	World Wide	Thomson Holidays Ltd	
B0007	Flight Only	Thomson Holidays Ltd	
B0008	A La Carte	Thomson Holidays Ltd	
B0009	Weddings In Paradise	Thomson Holidays Ltd	
B0010	All Inclusive	Thomson Holidays Ltd	
B0011	Summer Sun	Thomson Holidays Ltd	
B0012	Winter Sun	Thomson Holidays Ltd	
B0013	Airtours Summer Sun	Airtous Holidays Ltd	
B0014	Airtours Winter Sun	Airtous Holidays Ltd	
B0015	Airtours Far and Away	Airtous Holidays Ltd	
B0016	Weddings of Fantasy	Airtous Holidays Ltd	
B0017	All Inclusive	Airtous Holidays Ltd	
B0018	Airtours Flight Only	Airtous Holidays Ltd	
B0019	Airtours Florida	Airtous Holidays Ltd	
B0020	Airtours Cruises	Airtous Holidays Ltd	
B0021	Golden Years	Airtous Holidays Ltd	
B0022	Airtours Fly-Drive	Airtous Holidays Ltd	

SQL

The screenshot shows a window titled "SQL" with a text area containing the following SQL statement:

```

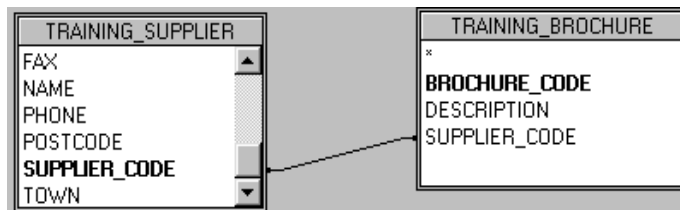
SELECT TRAINING_BROCHURE.BROCHURE_CODE,
TRAINING_BROCHURE.DESCRPTION,
TRAINING_SUPPLIER.NAME
FROM TRAINING_BROCHURE TRAINING_BROCHURE,
TRAINING_SUPPLIER TRAINING_SUPPLIER
WHERE TRAINING_SUPPLIER.SUPPLIER_CODE =
TRAINING_BROCHURE.SUPPLIER_CODE
  
```

Buttons for "OK" and "Cancel" are visible on the right side of the window.

TIP Microsoft Query executes the tables in alphabetical order, other applications such as Access execute the query with respect to the order the tables were added in. This could have implications for the optimisation of the query. Although not vital for this particular exercise, as we are retrieving information from the whole record, this would become important if we wanted to list the brochures for a particular supplier (see Task 9).

Task 9 : List all brochures for Thomson Holidays Ltd

Step One	Add the SUPPLIER table to the query
Step Two	Turn off Auto Query
Step Three	Add the NAME column to the query
Step Four	Add the BROCHURE table to the query
Step Five	Add the DESCRIPTION column to the query
Step Six	Ensure that the tables join on the correct fields (SUPPLIER_CODE)



Step Seven	Add a criterion to specify Thomson Holidays Ltd (since this is part of the key it can be accessed quickly. SUPPLIER_CODE could also be used).
------------	---



Criteria Field: NAME
 Value: Like 'Thomson%'
 or:

Step Eight	Run the query
------------	----------------------

Result Set

	NAME	DESCRIPTION
	Thomson Holidays Ltd	Faraway Shores
	Thomson Holidays Ltd	Skiing
	Thomson Holidays Ltd	Young At Heart
	Thomson Holidays Ltd	Price Breakers
	Thomson Holidays Ltd	City Breaks
	Thomson Holidays Ltd	World Wide
	Thomson Holidays Ltd	Flight Only
	Thomson Holidays Ltd	A La Carte
	Thomson Holidays Ltd	Weddings In Paradise
	Thomson Holidays Ltd	All Inclusive
	Thomson Holidays Ltd	Summer Sun
	Thomson Holidays Ltd	Winter Sun

SQL

```

SQL Statement:
SELECT TRAINING_SUPPLIER.NAME,
TRAINING_BROCHURE.DESCRPTION
FROM TRAINING_BROCHURE TRAINING_BROCHURE,
TRAINING_SUPPLIER TRAINING_SUPPLIER
WHERE TRAINING_BROCHURE.SUPPLIER_CODE =
TRAINING_SUPPLIER.SUPPLIER_CODE AND
((TRAINING_SUPPLIER.NAME Like 'Thomson%'))
  
```

SUGGESTED EXERCISE - MS ACCESS ONLY

Modify the order of the tables within the SQL query and see if you can spot the difference in time. (For the amount of data here the time is a matter of about 2 seconds, however if there were thousands of records then this would increase). Instead of reading the whole of BROCHURE and then finding the Supplier's name we would find the Supplier's name and read only those records within BROCHURE that matched our requirements.

TIP Optimisation would also increase if we used SUPPLIER_CODE instead of NAME as this is a unique key.

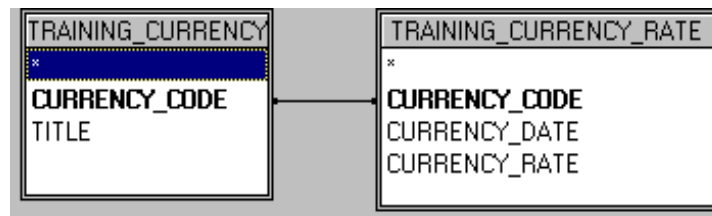
Task 10 : List existing currencies and currency rates

Note:

Details of currencies are in the CURRENCY_MASTER table

Currency rates are in the CURRENCY_RATE table

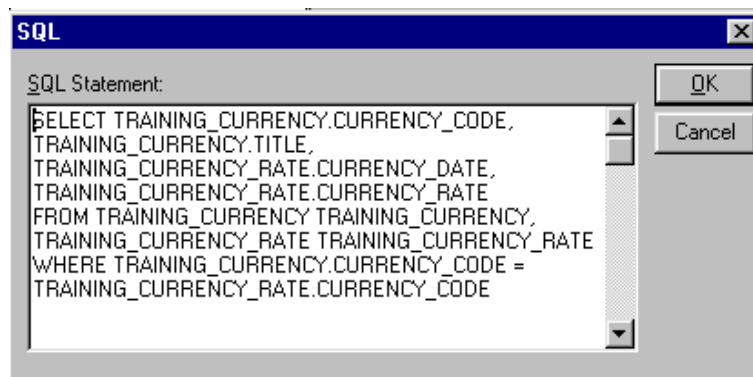
Step One	Add the CURRENCY_MASTER table to the query
Step Two	Turn off Auto Query
Step Three	Add the CURRENCY_CODE column to the query
Step Four	Add the TITLE column to the query
Step Five	Add the CURRENCY_RATE table to the query



Step Six	Add the CURRENCY_DATE column to the query
Step Seven	Add the CURRENCY_RATE column to the query
Step Eight	Run the query

Result Set

CURRENCY_CODE	TITLE	CURRENCY_DATE	CURRENCY_RATE
AED	United Arab Emirate Dirh	1997-01-01	162.
ALL	Albanian Lek	1997-01-01	203.172
ATS	Austrian Schilling	1997-01-01	22.26
AUD	Australian Dollar	1997-01-01	2.58
BDT	Bangladeshi Taka	1997-01-01	87.
BEF	Belgian Franc	1997-01-01	65.256
BGL	Bulgarian Lev	1997-01-01	969.384
BHD	Bahraini Dinarling	1997-01-01	0.6169
BIF	Burundi Franc	1997-01-01	624.108
BRL	Brazilian Real	1997-01-01	2.112
BWP	Botswana Pula	1997-01-01	7.404
CAD	Canadian Dollar	1997-01-01	2.808
CHF	Swiss Franc	1997-01-01	2.748
CLP	Chilean Peso	1997-01-01	859.872



SUGGESTED EXERCISE

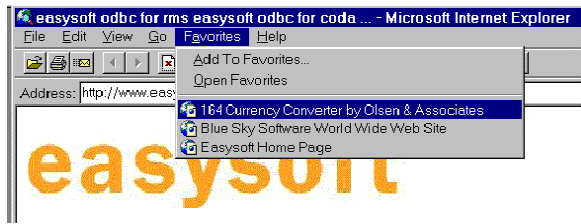
Translate currency rates from GBP to a currency of your choice, and update this information into RMS.

The remainder of this section gives step-by-step instructions for getting currency rates from the internet and for uploading the data.

1. Start a web browser of your choice. Here we use Microsoft Internet Explorer. To start this, double-click the icon or select the option from the **Start** menu.



2. Go to <http://www.oanda.com/cgi-bin/ncc>, the Olsen and Associates Currency Converter page. (A shortcut has been set up for you under **Favourites**).



3. Select the currencies to use and the date (the default date is today) and click the "Convert Now" button. (Full instructions taken from the Olsen web page are shown in the next step).



4. Here are the online instructions:

Follow these steps to convert currencies.

1. Click the "Currency Converters" folder in the SiteSeeing frame, then choose the language you want or click 164 Currencies Converter in the OANDA homepage. See [To find the 164 Currencies Converter in the SiteSeeing frame](#).
2. Choose a currency or precious metal you want to convert **from** in the left scrolling list. Click the scrolling arrows to see more currency selections.
3. Choose a currency or precious metal you want to convert **to** in the right scrolling list. Click the scrolling arrows to see more currency selections.
4. If you want to change the amount of the currency to convert **from**, click the text box next to "Convert Amount" and type the amount of the currency you want to convert. The default is "1."
5. Click the "Convert Now!" button.

Important: All currencies are listed with the name of their country first. For example, the "Peso" for Mexico is listed as "Mexican Peso."

5. An example currency conversion is shown here:

On Wednesday, July 9, 1997

1 British Pound = 0.9623 Latvian Lats
(1 Latvian Lats = 1.0392 British Pound)

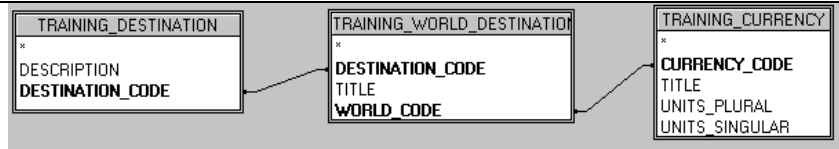
Median price was 0.9623 / 0.9638 (bid/ask).
Estimated price based on daily US dollar rates.

6. Now you have the new rate, you can update the RMS data. Since you cannot update to more than one table using Microsoft Query, remove the CURRENCY table from the download query you had been working on and refresh the data (or start a new query).
7. Switch on the option in Microsoft Query which allows editing (**Records, Allow Editing**).
8. Enter the currency code, date and rate for the currency you want to update.
9. Move the cursor off the row; the new data will be uploaded.

Task 11 : Find the currency used in Cancun

Note: The table WORLD_DESTINATION holds information on DESTINATION and WORLD_CODE. WORLD_CODE is based upon the currency code.

Step One	Add the tables DESTINATION, WORLD_DESTINATION and CURRENCY to the query
Step Two	Turn off Auto Query
Step Three	Ensure the tables are joined correctly



Step Four	Add criteria to select Cancun
Step Five	Select the columns DESCRIPTION, CURRENCY_CODE and TITLE (from the CURRENCY table)
Step Six	Run the query

Result Set

	DESCRIPTION	CURRENCY_CODE	TITLE
▶	Cancun	MXP	Mexican Pesco



```
SELECT TRAINING_DESTINATION.DESCRPTION,
TRAINING_CURRENCY.CURRENCY_CODE, TRAINING_CURRENCY.TITLE
```

```
FROM TRAINING_CURRENCY TRAINING_CURRENCY,
TRAINING_DESTINATION TRAINING_DESTINATION,
TRAINING_WORLD_DESTINATION TRAINING_WORLD_DESTINATION
```

WHERE TRAINING_WORLD_DESTINATION.DESTINATION_CODE =
TRAINING_DESTINATION.DESTINATION_CODE AND
TRAINING_WORLD_DESTINATION.WORLD_CODE =
TRAINING_CURRENCY.CURRENCY_CODE AND
((TRAINING_DESTINATION.DESCRPTION='Cancun'))

ADDITIONAL EXERCISE

Find the currency rate for Cancun for a particular date of your choice.

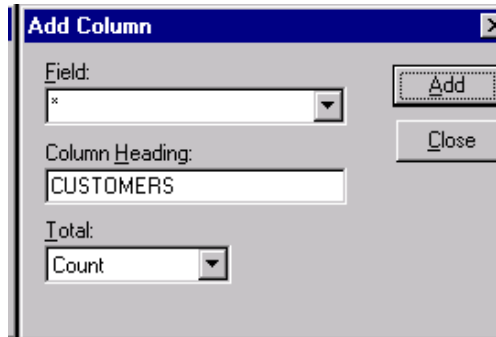
Example result set:

	DESCRIPTION	CURRENCY_C	TITLE	CURRENCY_D	CURRENCY_RATE
▶	Cancun	MXP	Mexican Peso	1997-01-01	15.876

Task 12 : How many customers are there?

Note: This can be achieved by counting the rows of the CUSTOMER table.

Step One	Add the CUSTOMER table to the query
Step Two	Select Records from the tool bar and choose Add Column . Complete the dialog box with the necessary information

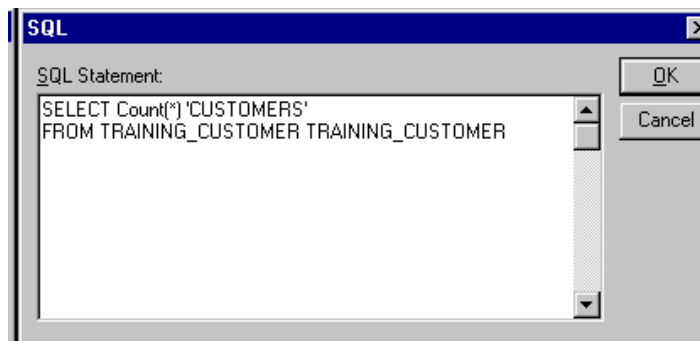


Step Four	Click Add . Then run the query
-----------	---------------------------------------

Result Set

CUSTOMERS
20

SQL



SUGGESTED EXERCISE

Try counting all customers with particular post code or living in a particular town by replacing * with the column name and relevant criteria.

Task 13 : Update the CUSTOMER file

Note: Mr Martin Smith's Address has changed from 38 Watergate Way, Harehills, Leeds, West Yorkshire, LS3 3BJ to 50 Hanover Lane, Lowfold, Yeadon, West Yorkshire, LS23 7GB. His telephone number stays the same. Update the CUSTOMER file to reflect this.

Step One	Add the CUSTOMER table to the query
Step Two	Turn off Auto Query
Step Three	Add the ADDRESS1, ADDRESS2, ADDRESS3, TOWN and POSTCODE columns to the query, plus any other relevant columns
Step Four	Add criteria to select only data where NAME equals Smith and POSTCODE equals LS3 3BJ. Or simply use CUSTOMER_CODE equal to C0011



Criteria Field:	NAME	POSTCODE
Value:	'Smith'	'LS3 3BJ'
or:		

Step Five	Modify the appropriate columns for the new address
Step Six	Move the cursor off the row to upload the changes to the sever files
Step Seven	Modify the query to display Mr Martin Smith's new address



It is not possible to see the SQL produced from within Query. The SQL for the update only would be:

```
UPDATE TRAINING_CUSTOMER
SET TRAINING_CUSTOMER.ADDRESS1 = '50 Hanover Lane',
TRAINING_CUSTOMER.ADDRESS2 = 'Lowfold',
TRAINING_CUSTOMER.TOWN = 'Yeadon',
TRAINING_CUSTOMER.POSTCODE = 'LS23 7GB'
WHERE (TRAINING_CUSTOMER.CUSTOMER_CODE='C0011')
```

Task 14 : Insert a New Supplier Record

Note: The company name and address details to insert are as follows: Avro Holidays, 1st Floor, Wavell House, Holcombe Road, Helmshore, Lancashire, BB4 4BN, 01706 424242.

Step One	Add the SUPPLIER table to the query
Step Two	Turn off Auto Query
Step Three	Select all columns (*) or SUPPLIER_CODE, NAME, ADDRESS1, ADDRESS2, ADDRESS3, TOWN, COUNTY, POSTCODE, PHONE
Step Four	Click Run Query
Step Five	Click Records on the toolbar and select Allow Editing . (If it is already selected it a tick will appear next to the selection) This adds a blank row onto the bottom of the table
Step Six	Type in the relevant supplier code, name and address details
Step Seven	Move the cursor off the row to insert the new record



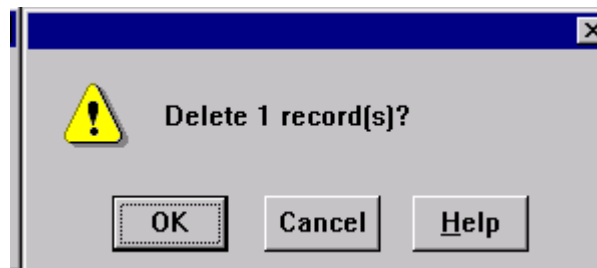
It is not possible to see the SQL produced from within Query. The SQL for the insert only would be:

```
INSERT INTO TRAINING_SUPPLIER (SUPPLIER_CODE, NAME, ADDRESS1,
ADDRESS2, ADDRESS3, TOWN, COUNTY, POSTCODE, PHONE) VALUES
('S0014', 'Avro Holidays', '1st Floor', 'Wavell House', 'Holcombe Road', 'Helmshore',
'Lancashire', 'B4 4BN', '1706 424242')
```

Task 15 : Delete a Supplier record

Note: The company to delete is Avro Holidays.

Step One	Add the SUPPLIER table to the query
Step Two	Turn off Auto Query
Step Three	Select all columns (*) or SUPPLIER_CODE along with any other columns
Step Four	Click Records on the toolbar and select Allow Editing . (If it is already selected it a tick will appear next to the selection) This adds a blank row onto the bottom of the table
Step Five	Add criteria to select only the supplier record to be deleted (e.g. by SUPPLIER_CODE or NAME)
Step Six	Highlight the row and click Delete to delete the record. A warning message will appear (see below). Click OK to delete the record permanently



It is not possible to see the SQL produced from within Query. The SQL for the Delete only would be:

```
DELETE FROM TRAINING_SUPPLIER WHERE SUPPLIER_CODE='S0014'
```

Note : If the WHERE clause were not included here, the whole of the SUPPLIER table would be deleted.

Task 16 : Find the Holiday company we have spent the most money with

Note: To carry out this task, you need to write multiple queries; this will have to be completed within Microsoft Access.

Query1 : Finds out the maximum Total Invoice Value of all Purchase Invoices

Query2 : Queries the Invoice Values and Supplier codes for all Purchase Invoices.

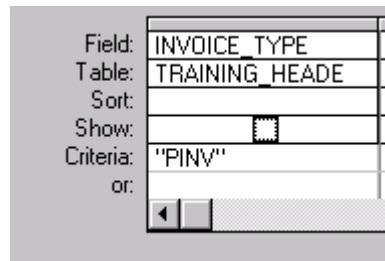
Query3 : Links Query1 and Query2 together to get the corresponding code for the maximum Invoice Value. Also, finds out the title of that code i.e. The company name.

The query has to be written in this way due to the use of aggregate functions (e.g. MAX, MIN, etc.).

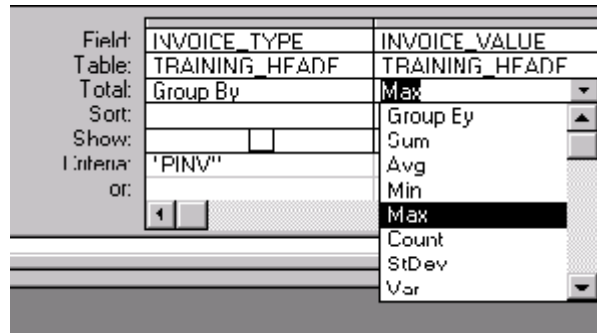
Query1

This can be done by evaluating the Total Values of Purchase Invoices, in this instance this information is contained in the Invoice Header.

Step One	Attach the tables HEADER and SUPPLIER
Step Two	Choose New Query within Design View
Step Three	Add the HEADER table to the query
Step Four	Add criteria to select only Purchase Invoices; do not display this column (deselect the Show option to hide the column - see below)



Step Five	Select View from the toolbar menu and click Totals (A tick should appear next to the option)
Step Six	Add the column INVOICE_VALUE to the query. From the Total line select the Max. option. This will automatically insert a GROUP BY next to the INVOICE_TYPE

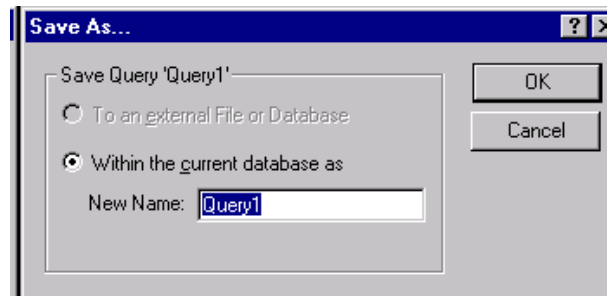


Step Seven	Run the query
------------	---------------

Result Set

MaxOfINVOICE_VALUE
13989

Step Eight	Select File and Save As to save the query within the current database
------------	---



Query2

Step Nine	Open a New Query in Design View
Step Ten	Add the HEADER table to the query
Step Eleven	Select INVOICE_TYPE, INVOICE_VALUE and CODE
Step Twelve	Add criteria to select only Purchase Invoices (PINV), hide this column
Step Thirteen	Run the query and Save it within the current database
Step Fourteen	Open a new query in Design View

Query3

Step Fifteen	Add Query1 and Query2 into the query (select Queries instead of tables from the Show Table box)
Step Sixteen	Join the two queries by Maximum Invoice Value and Invoice Value



Step Seventeen	Add the INVOICE_VALUE and CODE columns to the query
Step Eighteen	Add the SUPPLIER table to the query. Join CODE and SUPPLIER_CODE and select the column NAME
Step Nineteen	Run the query

Result Set

	INVOICE_VALUE	CODE	NAME
▶	13989	S0001	Thomson Holidays Ltd

SUGGESTED EXERCISES

- i. Find the customer who has spent the most money (i.e. Look at Sales Invoices (SINV)).
- ii. Find the company we have spent the least amount of money with (use the SQL MIN function).
- iii. Find the customer who has spent the least amount of money.

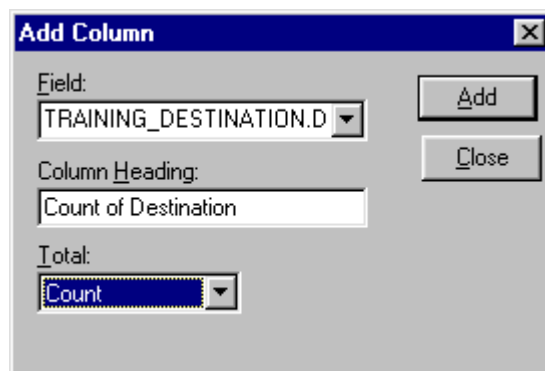
Result Set for exercise 1


	INVOICE_VALUE	CODE	TITLE	FORENAME	NAME
▶	8723.55	C0007	Miss	Keely	Gothard

Task 17 : Find the most popular holiday destination

Note: The one sold the most.

Step One	Add the HEADER table to the query
Step Two	Turn off Auto Query
Step Three	Add criteria for Sales Invoices (INVOICE_TYPE = SINV)
Step Four	Add the DETAILS table to the query. Ensure the join is on INVOICE_NUMBER
Step Five	Select Record and Add Column to find the Count of DESTINATION_CODE (see below). Click Add



Step Six	Add DESTINATION_CODE to the query (this will cause the above Count to GROUP BY DESTINATION_CODE)
Step Seven	Add the DESTINATION table to the query and select the Column DESTINATION_CODE
Step Eight	Click OK on the SQL when you are happy with it to run the query
Step Nine	We need to order by descending order to see which is the most popular this cannot be done in Microsoft Query itself when using a COUNT. So therefore the Data has to be pulled back into Excel. Select File and Return Data to Microsoft Excel.
Step Ten	Highlight all the columns and click  to sort in descending order. You will easily be able to see the most popular destination (see Result Set)

Result Set

3	D0089	Honolulu
2	D0007	Calais
2	D0012	Venice
2	D0013	Sorrento
2	D0022	Playa De Las Americas
2	D0066	Salzburg
2	D0077	Clearwater Bay
2	D0086	Montego Bay
1	D0003	Edinburgh
1	D0004	Blackpool
1	D0010	Rome
1	D0014	Lake Como

SQL

```

SELECT Count(TRAINING_DETAILS.DESTINATION_CODE),
TRAINING_DETAILS.DESTINATION_CODE, TRAINING_DESTINATION.DESCRPTION
FROM TRAINING_HEADER, TRAINING_DETAILS, TRAINING_DESTINATION
WHERE TRAINING_DESTINATION.DESTINATION_CODE =
TRAINING_DETAILS.DESTINATION_CODE
AND TRAINING_HEADER.INVOICE_NUMBER =
TRAINING_DETAILS.INVOICE_NUMBER AND
((TRAINING_HEADER.INVOICE_TYPE='SINV'))
GROUP BY TRAINING_DETAILS.DESTINATION_CODE,
TRAINING_DESTINATION.DESCRPTION

```

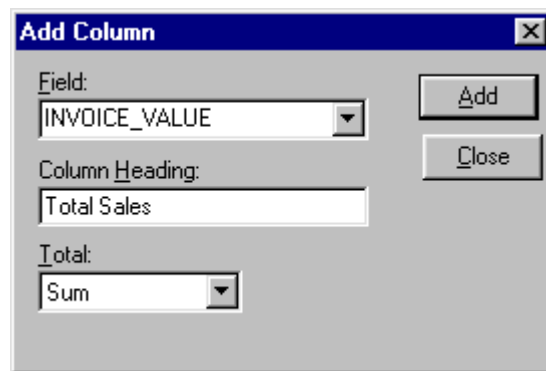
SUGGESTED EXERCISE

- i. Find the least popular sold holiday destinations.

Task 18 : Prepare a report to show the Total Profits

Note: This is simply a calculation of Sales-Purchases = Profits.

Step One	Add the HEADER table to the query
Step Two	Turn off Auto Query
Step Three	Add a criterion to specify Sales Invoices (INVOICE_TYPE=SINV)
Step Four	Select Records , Add Column from the toolbar. Make a Sum of INVOICE_VALUE



Step Five	Run the query to see the total value of all sales. Select File and Save Query from the toolbar menu. Call the query sales.qry
Step Six	Select File and Return Data to Microsoft Excel
Step Seven	Select Data and Get External Data from the toolbar menu. When you are presented with the Data source Dialog box click Cancel and select File Open Query from the toolbar. Open up the previously saved sales.qry and modify the criterion to select Purchase Invoices (INVOICE_TYPE= PINV). Save this query as purchase.qry.
Step Eight	Run the query and return the data to Microsoft Excel
Step Nine	Calculate Sales - Purchases to Give the Total Profit

Total Sales
£63,250.55
Total Purchases
£57,500.50
TOTAL PROFIT
£5,750.05

Task 19 : Prepare a letter to all Clients who spent over £5000.00 in May

Note: Complete this task using Microsoft Word Mail Merge. The letter is to offer clients £50.00 off their next holiday if it is booked before the end of December 1997. Follow the steps provided in the module entitled "Using ODBC Applications" to open Microsoft Query from within Word.

Step One	Add the HEADER table to the query
Step Two	Turn off Auto Query
Step Three	Add criteria to specify Sales Invoices (INVOICE_TYPE=SINV), Invoice date in May (INVOICE_DATE can either use >= or <= or BETWEEN) and Invoice Value is greater than £5000.00

Criteria Field:	INVOICE_TYPE	INVOICE_DATE	INVOICE_VALUE
Value:	'SINV'	>=#01/05/97#	>5000
or:		<=#31/05/97#	>5000


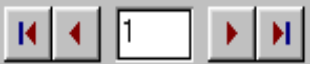
Step Four	Add the CUSTOMER table to the query and join HEADER.CODE to CUSTOMER.CUSTOMER_CODE. (Drag the mouse on the relevant fields to make a join between the two tables, see below)
-----------	--

<table border="1"> <tr><th>TRAINING_HEADER</th></tr> <tr><td>CODE</td></tr> <tr><td>CURRENCY</td></tr> <tr><td>INVOICE_DATE</td></tr> <tr><td>INVOICE_NUMBER</td></tr> <tr><td>INVOICE_TYPE</td></tr> </table>	TRAINING_HEADER	CODE	CURRENCY	INVOICE_DATE	INVOICE_NUMBER	INVOICE_TYPE	<table border="1"> <tr><th>TRAINING_CUSTOMER</th></tr> <tr><td>ADDRESS3</td></tr> <tr><td>COMMENTS</td></tr> <tr><td>COUNTY</td></tr> <tr><td>CUSTOMER_CODE</td></tr> <tr><td>FAX</td></tr> <tr><td>FORENAME</td></tr> </table>	TRAINING_CUSTOMER	ADDRESS3	COMMENTS	COUNTY	CUSTOMER_CODE	FAX	FORENAME
TRAINING_HEADER														
CODE														
CURRENCY														
INVOICE_DATE														
INVOICE_NUMBER														
INVOICE_TYPE														
TRAINING_CUSTOMER														
ADDRESS3														
COMMENTS														
COUNTY														
CUSTOMER_CODE														
FAX														
FORENAME														

Step Six	Select the columns you want to see in your letter (e.g. NAME, ADDRESS, etc.)
Step Seven	Run the query

Result Set

	TITLE	FORENAME	NAME	ADDRESS1	ADDRESS2
	Mr	Jason	Crummack	66 Collingway Close	
	Miss	Keely	Gothard	44 Park Road West	Sherburn

Step Eight	Return the results to Microsoft Word
Step Nine	<p>Insert the Merge Fields into the document.</p>  <p>view the data</p>  <p>scroll buttons</p>

The document could look like the following

Mr Jason Crummack
66 Collingway Close
Pontefract
West Yorkshire
WF7 6HT

CONGRATULATIONS

Dear Mr Crummack

We are pleased to inform you that you are now an Easygo Gold customer and have won a voucher entitles you to **£50.00** yes **£50.00 off** your next holiday* booked before 31st December 1997.

Simply present the enclosed voucher to one of our friendly staff and you will receive **£50.00**

Task 20 : Produce a Purchase Invoice form

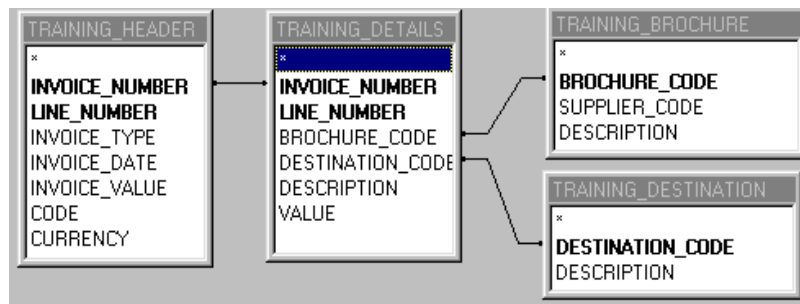
Note: Complete this task using Microsoft Access. For the purpose of this exercise we will use the Form Wizard as a quick way of formatting the form. The form will contain a subform.

Step One	Attach all of the tables to Microsoft Access. (The easiest way is to right click with the mouse)
Step Two	Select to create a New Query in Design View
Step Three	Add the HEADER table to the query and Add criterion to select only Purchase Invoices (INVOICE_TYPE=PINV)
Step Four	Add the SUPPLIER table to the query and join HEADER.CODE to SUPPLIER.SUPPLIER_CODE to enable us to obtain the Name and Address Details of the Supplier. (Drag the mouse on the relevant fields to make a join between the two tables, see below)



Step Five	Select the columns you want to see on the top of your Invoice (e.g. Invoice Number, Value, Name, Address, etc.)
Step Six	Run the query Select File from the Tool Bar and then Save As/Export and save a copy of the query within the Current Database (e.g. Header)
Step Seven	Add a new Query and add the HEADER table and the DETAILS table to the query make a Join on INVOICE_NUMBER between the HEADER and DETAILS table. Add a criterion to select Purchase Invoices (PINV). This will enable you to see all of the details line for each Purchase Invoice Number
Step Eight	Add the BROCHURE and DESTINATION tables to the query. These will automatically be linked with BROCHURE_CODE and DESTINATION_CODE from the DETAILS table to enable you to

	retrieve a description of the codes
--	-------------------------------------

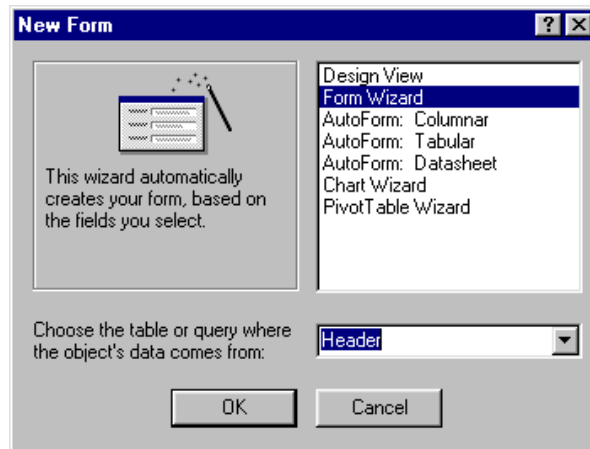


Step Nine	Select the columns which describe the Codes included in the Detail lines of the Invoice, namely all of the Descriptions plus the Value. INVOICE_NUMBER from the DETAILS table will also need to be included to enable us to design a form. (BROCHURE_DESCRIPTION, DESTINATION_DESCRIPTION, DETAILS_DESCRIPTION)
Step Ten	Run the query

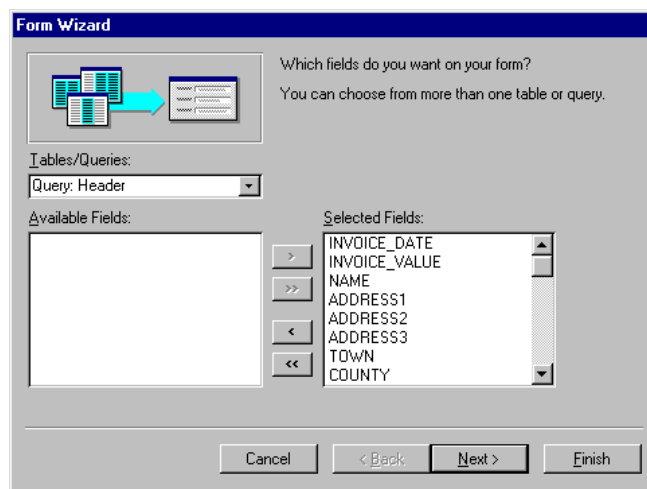
Result Set

INVOICE_NUM	TRAINING_BR	TRAINING_DE	TRAINING_DETAILS.DESCRPTION
2000	Faraway Shore	Honolulu	2 Passengers for 2 Weeks
2000	Ski-ing	Salzburg	4 Passengers for 1 Week
2000	Young At Heart	Benidorm	2 Passengers for 21 nights
2000	Price Breakers	Playa De Las A	8 Pax 1 Apt 14 Nights
2000	City Breaks	Venice	Honeymoon- Luxury Double 5 nights
2000	World Wide	Hong Kong	Flight Only Economy 2 people 1 infant
2000	Summer Sun	Clearwater Bay	14 Nights Family of 4
2000	Summer Sun	Torremolinos	10 nights 4 people
2000	Summer Sun	Playa De Las A	3 Weeks Flight Only
2001	All Inclusive	Montego Bay	2 Weeks All Inclusive
2001	Airtours Cruises	Miami	2 Passengers 5 night special cruise
2001	Airtours Florida	Clearwater Bay	Family of 4 Fly-Drive 14 Nights

Step Eleven	Select File from the Tool Bar and then Save As/Export and save a copy of the query within the Current Database (e.g. Details)
Step Twelve	Select Form, New . You are presented with a dialog box, see below. Select to use the form Wizard. Here we will add a form and a sub form within that form. The form itself will be based around the Header Query created at the beginning of this task



Step Thirteen	Make sure you are using the correct query to create the form (Tables/Queries) and select the fields you wish to include within the form. > for individual fields or >> for all fields. Click Next once you are happy with your selection
------------------	---




Step Fourteen	Select the Layout of your form, the best option is Columnar. Click Next
Step Fifteen	Next you are prompted to select the Style of your form. In this example we use Stone. Click Next
Step Sixteen	Enter the Title for your form, and select the option to modify the form's design. Click Finish

Form Header	
Detail	
INVOICE_DATE	INVOICE_D
INVOICE_VALUE	INVOICE_VALUE
NAME	NAME
ADDRESS1	ADDRESS1
ADDRESS2	ADDRESS2
ADDRESS3	ADDRESS3
TOWN	TOWN
COUNTY	COUNTY
POSTCODE	POSTCODE
PHONE	PHONE
INVOICE_NUMBER	INVOICE_NL

Step Seventeen You can select the fields to reposition them by clicking on the left hand boxes a small hand will appear that will allow you to drag the form and drop it anywhere. You may also wish to resize the form (you can click on the line and drag it to enlarge it in any direction) and rename the column headings

Form Header	
Detail	
Invoice Date	INVOICE_D
Invoice Number	INVOICE_NL
Invoice Value	INVOICE_VALUE
Name	NAME
Address	ADDRESS1
	ADDRESS2
	ADDRESS3
	TOWN
	COUNTY
	POSTCODE

Step Eighteen Click on the Subform/subreport option from the toolbar  and draw a box underneath your header information where you would like to see the detail lines appear. This will take you into a subform/subreport wizard. Select the option to build a new subform/subreport from a Table/Query. Click **Next** and select the 2nd query saved in the previous tasks. Add the fields you wish to display in the subform.

Note INVOICE_NUMBER is to be included but can be hidden later to enable a link to be made between the form (header) and the subform (details)

Step Nineteen Access should automatically prompt you to make a link on INVOICE_NUMBER. Click **Finish** to return to design mode

Step Twenty	Select the Subform and double click with the mouse within the subform to edit any of the properties, delete the fields you do not want to appear. Edit the name of the field by right clicking on the data field within the design view
Step Twenty	Select View, Form to view the form

Final Result

The screenshot shows a Microsoft Access form with the following fields:

- Invoice Date:** 30/05/97
- Invoice Number:** 2000
- Invoice Value:** 13989
- Name:** Thomson Holidays Ltd
- Address:** Greater London House, Hamstead Road, London, Greater London, WC1 7SD
- Phone Number:** 0990 502399

Below the form fields is a table titled "Details subform" with the following data:

Brochure	Destination	Description	VALUE
Faraway Shores	Honolulu	2 Passengers for 2 Weeks	1798
Ski-ing	Salzburg	4 Passengers for 1 Week	1400
Young At Heart	Benidorm	2 Passengers for 21 nights	899
Price Breakers	Playa De Las Americas	8 Pax 1 Apt 14 Nights	2792
City Breaks	Venice	Honeymoon- Luxury Double 5 nights	910
World Wide	Hong Kong	Flight Only Economy 2 people 1 infant	1799
Summer Sun	Clearwater Bay	14 Nights Family of 4	2196
Summer Sun	Torremolinos	10 nights 4 people	1996
Summer Sun	Playa De Las Americas	3 Weeks Flight Only	199

You can scroll through the invoices by click the next record button at the bottom of the form.

SUGGESTED EXERCISE

i. Produce a Sales Invoice form.

Tip Copy the Details query already prepared for the Purchase Invoice and change the Document Type to be Sales Invoice (SINV)

Invoice Number Mr

Invoice Date

Invoice Value

SalesDetails subform

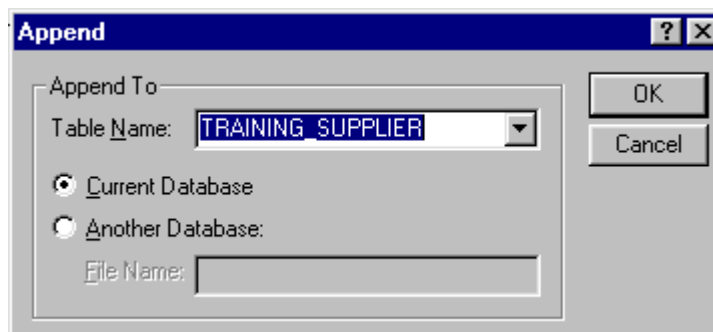
	Brochure	Destination	Description	VALUE
▶	Cosmos Sandals	Montego Bay	Honeymoon	6598.9
	P&O Ferries	Calais	1 car	152.9

Task 21 : Insert data into an RMS file from local Access Tables

Note: For the purpose of this exercise two local tables containing data (CONTACTS and SUPPLIERS) have been created within Access. This exercise will allow us to update an RMS data file on the server with information from the two tables.

This has to be done in two Steps. Firstly, the two local access tables need to be joined together. Secondly, an append query has to be written to insert the records into the file.

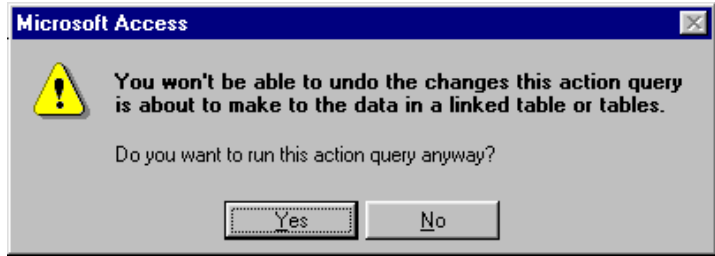
Step One	Open a new query in Access in Design View
Step Two	Add the local copies of the SUPPLIERS and CONTACTS table to the query. Ensure the tables are joined on their common fields, in this case SUPPLIER_CODE
Step Three	Select all of the columns into the query you would wish to insert into the RMS file on the server (SUPPLIER.DAT)
Step Four	Run the query to ensure everything is OK. You should see information about the suppliers and contacts
Step Five	Select Query from the toolbar menu and then Append. The append dialog box should appear (see below). Select the table name you wish to append to from the pull down box, i.e. TRAINING_SUPPLIER. Select the Current Database. Click OK .



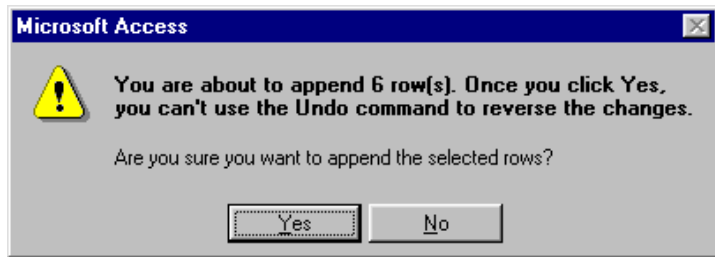
Step Six	This will add a new line (Append To) within the design of your query (see below). This will contain details of the matching columns. If the column names are different you can select from the pull down options the column you want to append to. In this case we want to append SUPPLIERCODE to be SUPPLIER_CODE. Also some contact details are named differently
----------	---

Field:	SUPPLIERCODE	NAME	ADDRESS1	ADDRESS2	ADDRESS3
Table:	SUPPLIER	SUPPLIER	SUPPLIER	SUPPLIER	SUPPLIER
Sort:					
Append To:		NAME	ADDRESS1	ADDRESS2	ADDRESS3
Criteria:	TRAINING_SUPPL				
or:	SUPPLIER_CODE				
	NAME				
	ADDRESS1				
	ADDRESS2				

Step Seven When you are sure that the columns match up run the query. The following message will appear. Click **Yes**



Step Eight Another message box will appear advising you how many rows you are to append. Click **Yes**. This will now have inserted the records into the file

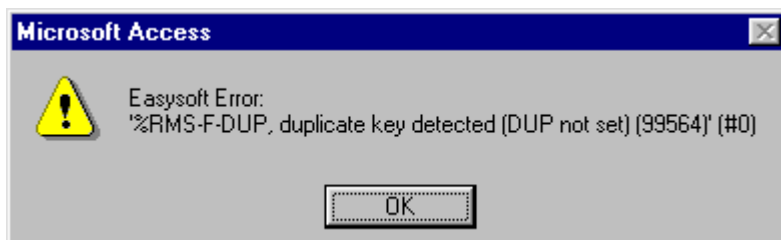
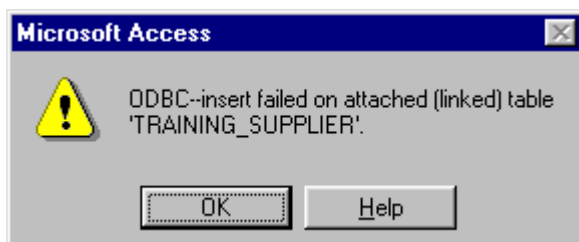


Step Nine Attach the TRAINING_SUPPLIER table and run a query to look at the new records

Result Set

	SUPPLIER_CO	NAME	ADDRESS1	ADDRESS2	ADDRESS3	TOWN	COUNTY
	S0014	Unijet	Unijet House	Second Street		Bristol	
	S0015	Easyjet	Sovereign Street			Leeds	
	S0016	International To	International Ho	Park Row		Leeds	West Yorkshir
	S0017	Kenmagara	13 Town Street			Brighouse	West Yorkshir
	S0018	Butlins Holiday:	The Booking Ce	117 Highmore L	Kensington	London	Greater Londo
	S0019	Pontins	Pontins House	238 Manningha		Bradford	West Yorkshir
*							

Note: If you try to insert the same data into a table you would see the following error messages.



12. Easysoft Excel Macro for RMS

In this module you will learn what the macro is and how to

- ✦ install the macro
- ✦ initialise the macro
- ✦ download data and set selection criteria
- ✦ order data
- ✦ use the pivot table option
- ✦ upload data to the server
- ✦ create and run reports
- ✦ use Excel cell references to create a generic report

Contents

Overview of Macro	12-2
Macro Installation	12-3
Initialisation	12-5
Download Data	12-7
Complex Criteria	12-11
Pivot Table	12-13
Upload Data	12-16
Report	12-18
Setup New Report	12-18
Run Report	12-19
Advanced Techniques	12-19
Troubleshooting	12-22

Overview of Macro

The Easysoft Excel Macro for RMS (also known as the “RMS macro”) allows you to download data, create reports and upload data to RMS with the minimum of effort.

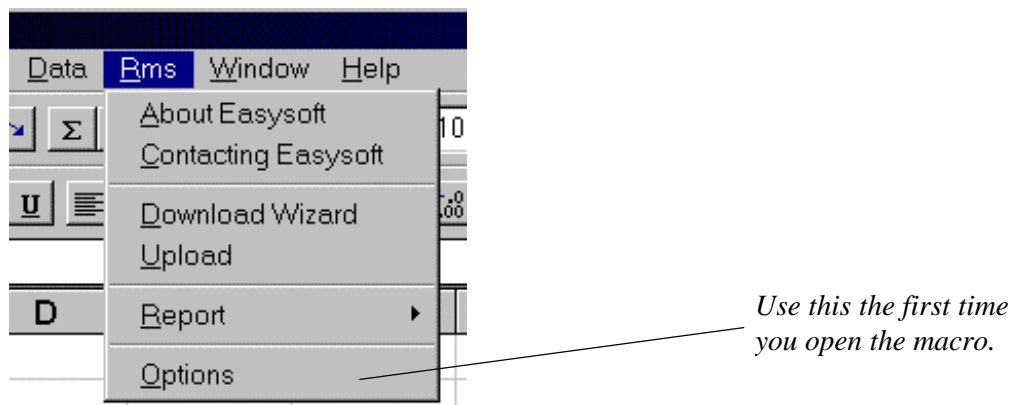
There are two versions of the Easysoft Excel Macro for RMS.

16 bit	RMS16.XLA	for Microsoft Excel version 5.x only
32 bit	RMS32.XLA	for Microsoft Excel 95 and Excel 97

The software applies to PC versions of Excel only.

The Easysoft Excel Macro for RMS requires Easysoft ODBC for RMS version 1.3 build 250 and above on both the client and the server.

The **Rms** menu option on the menu bar gives access to the top level pulldown shown here.



Before the macro can be used, it must be installed, and this is described in the next section. Then the following topics are discussed: downloading data, uploading data, creating reports and finally, troubleshooting.

Use the **About Easysoft** option to obtain version information about the macro.

Use the **Contacting Easysoft** option to list the support contact options.

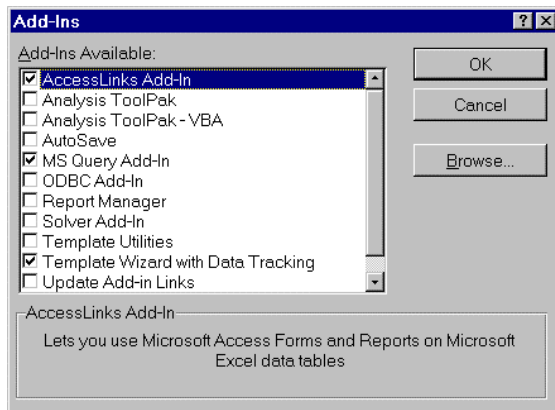
Macro Installation

To install the macro, you must add the software as an Excel add-in. Take the following steps:

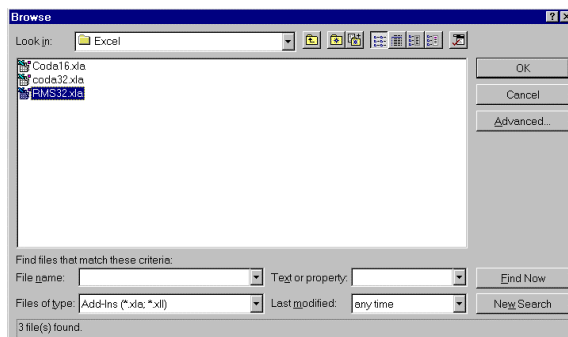
1. Copy the macro to a convenient location (the recommended location is `c:\easysoft\excel`).

If you obtain the software from the Easysoft web site your web browser should allow you to download and save the file in the newly created directory. If you obtain the software on floppy disk, copy it to the directory.

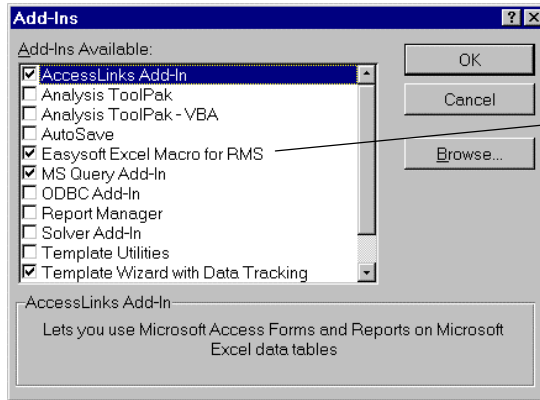
2. Start Microsoft Excel. Select **Add-Ins...** from the **Tools** menu. The Add-Ins dialog box appears.



3. Click the **Browse...** button, navigate to the `c:\easysoft\excel` directory, and highlight the `.XLA` file. Click the **OK** button.



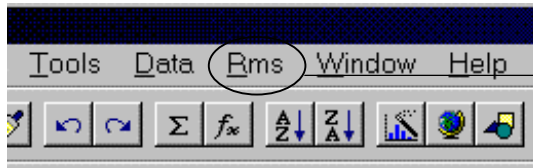
4. The Add-in now appears in the Add-Ins Available list box.



The Easysoft macro is now accessible.

Select the RMS add-in, and click the **OK** button.

5. The RMS macro is now available for use by clicking the newly added **Rms** menu which appears on the Excel menu bar.



*New **Rms** option*

Upgrading the Macro

To upgrade the macro, overwrite the existing *.XLA file(s) for the macro with the new one(s).

Initialisation

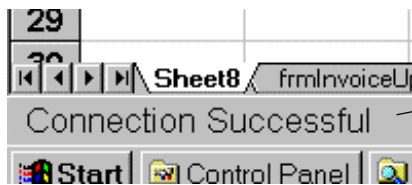
The first time the macro is used on a PC this initialisation step must be completed. It only has to be done once (unless you want to change the options); the information is saved, and when you, or another user, uses the macro in future, the information is already available for use.

1. To set the initialisation options, select **Options** from the **Rms** menu. The Options dialog box appears. The various settings you can choose are described in the remainder of this section.



2. **Datasource:** select the required data source from the dropdown list box (in these examples we use Training Data). Only data sources which use Easysoft drivers are shown in the dropdown list.
3. **Query Timeout** refers to the maximum length of (clock) time that the query is allowed to take up. If the query is not completed within this time, it will be cancelled. The default value of 600 seconds can be changed if required.
4. **Limit Rows** is used to limit the number of rows initially returned by the query. If the query result contains more rows than the number specified, you will be asked whether or not you wish to see these. The default value of 300 rows can be changed if required.
5. **Lookup Rows** is not enabled in this version of the macro. The number in the box has no effect on the operation of the macro.
6. **Keep Connection Open** is used to keep the connection to the data source in an open state. This can speed up the accessing of data, but the trade-off is that if many users want to access the RMS data, there may not be enough licence slots available. By default, this option is selected.

7. **Enable Status Message** is used to show messages at the bottom of the worksheet e.g.



This is where you will see messages.

By default, this option is selected.

8. **Auto Skip Failed Updates** is used to prevent on-screen error messages which result from attempts at invalid updates being shown. However, at the end of the update operation, all rows which the user wanted to update, but which were not updated will be flagged on the worksheet (see “Note - Auto Skip Failed Updates”, page 12-17). By default, this option is selected.
9. **Debug Message Trace.** If problems are being experienced, this option should be selected. It results in messages being displayed on screen. By default, this option is not selected.
10. When all the information has been entered, click the **OK** button to accept it. Control passes back to Microsoft Excel, and you can now use the macro functions, which are described in the following sections.

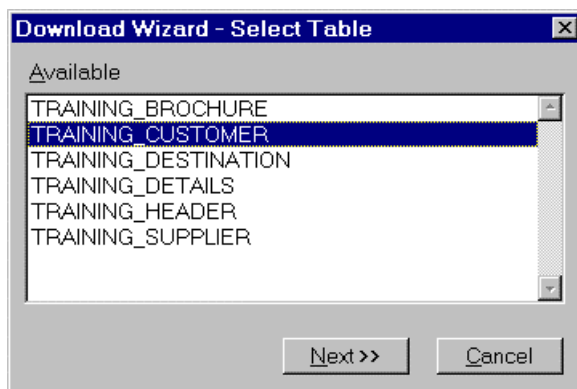
Or click the **Cancel** button to return to Excel without saving any changes you may have made. If you have not previously initialised the macro, you will not be able to use it.

Reset is not enabled in this version of the macro. Clicking the button has no effect on the operation of the macro.

Download Data

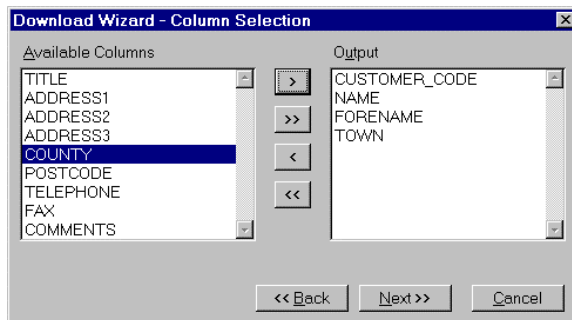
The Download Wizard consists of a series of dialog boxes which you should work through in sequence. When you have completed each dialog box, you should click the **Next>>** button to move onto the next one. If you change your mind about an earlier entry, then click the <<**Back** button(s) to move back to an earlier dialog box. This step-by-step guide shows you then entire process. At the end of this section there are explanations of the more advanced options which are available.

1. To download data, select **Rms, Download Wizard**. You will be presented with the Table Download dialog box. In this example, we will download customer information.



Select the table you wish to download. Then click the **Next>>** button.

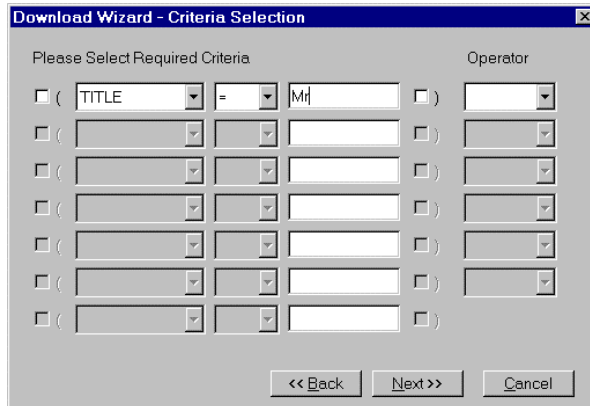
2. The Column Selection dialog box appears. From the Available Columns list, highlight each column you wish to see, then click the **>** button (you cannot select multiple columns). If you want to see all the columns, then just click **>>** without highlighting them.



The column(s) will be moved to the Output box. Click the **Next>>** button.

3. The Criteria Selection dialog box appears. Enter the criteria you require (if any), then click the **Next>>** button (complex criteria are discussed on page 12-11). Here we select customers with a title of “Mr”. Columns in the selection criteria do not need to

have been chosen in the Column Selection dialog box.



- The Sort Order dialog box appears. Use this to order the data that is retrieved (this is optional; you do not have to set a sort order). You can select up to 3 columns on which to order the result. For each column, the sort order may be ascending or descending. The default ordering is ascending, and is not indicated. To sort data in descending order, click the check box to the right of the column name. Click the **Next>>** button when you have made your selection.

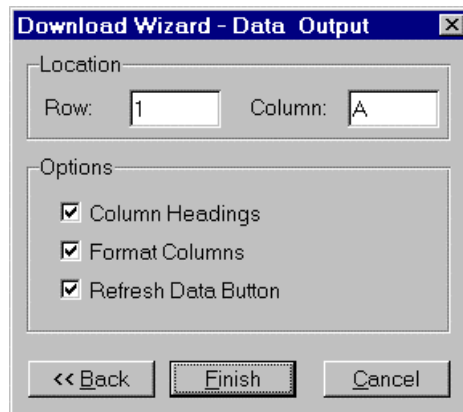


- The Output Destination dialog box appears. By default, the data will be sent to a new worksheet - the worksheet will be selected automatically. If you want to send the data to an existing worksheet, then select that option, and select the name of the worksheet from the dropdown dialog box.

You can either output the data directly to the worksheet (this is the default option) or you can create a pivot table (see page 12-13). Click the **Next>>** button when you have made your selection.



6. Assuming that you selected the output type “Data” in the previous dialog box, the Data Output dialog box appears. This controls the location of the data that is downloaded. Row specifies the first row on the worksheet in which the data will be placed, and Column specifies the first column in which the data will be placed.



Column Headings: specifies whether or not the names of the columns will be shown on the worksheet. By default, this option is selected.

Format Columns: sets the widths of the columns which contain downloaded data so that the data can be easily seen. Also, if the Column Headings option is selected, then this formatting option sets colour of the cells. By default, this option is selected.

Refresh Data Button: places a button (named Refresh) on the worksheet. The button is created in the second empty column after the last column of data. Clicking this button after the data has been downloaded will refresh the data (that is, it will download the data again, according to the options you selected when you first downloaded it). Thus, if the data changes frequently, you can always be sure of having the latest version, without having to work through the entire Download Wizard each time you want the latest information. By default, this option is selected.

Click the **Finish** button to download the data.

	A	B	C	D	E	F
1	CUSTOMER_CODE	NAME	FORENAME	TOWN		Refresh
2	C0010	Unwalla	Mike	Sheffield		
3	C0015	Spencer	Richard	Bradford		
4	C0005	Smith	Guy	Wetherby		
5	C0011	Smith	Martin	Leeds		
6	C0008	Naylor	Ian	Rochdale		
7	C0017	Morley	Mark	Leeds		
8	C0002	Kos	John	Wakefield		
9	C0009	Gorman	Nick	Halifax		
10	C0013	Gillard	Simon	Halifax		
11	C0003	Crummack	Jason	Pontefract		

*Click the **Refresh** button to ensure data is up to date*

Note: the validation of the SQL that is generated from the dialog boxes in the Wizard does not occur until after the **Finish** button on the Data Output dialog box has been clicked.

Complex Criteria

You can set complex criteria using the Criteria Selection dialog box. The columns that you use do not need to appear in the output columns. This section shows you how to set two criteria, with an emphasis on the correct use of the bracketing options. You will see how this affects the results in queries containing the SQL OR operator.

As a basis for the queries, we will use the CUSTOMER table; for convenience, some of the columns are replicated here.

CUSTOMER_CODE	NAME	FORENAME	TITLE	TOWN
C0001	Richardson	Carolyn	Miss	Brighouse
C0002	Kos	John	Mr	Wakefield
C0003	Crummack	Jason	Mr	Pontefract
C0004	Welburn	Caroline	Miss	Wakefield
C0005	Smith	Guy	Mr	Wetherby
C0006	Berlin	Karl	Mr	Leeds
C0007	Gothard	Keely	Miss	Leeds
C0008	Naylor	Ian	Mr	Rochdale
C0009	Gorman	Nick	Mr	Halifax
C0010	Unwalla	Mike	Mr	Sheffield
C0011	Smith	Martin	Mr	Leeds
C0012	Carter	Michael	Mr	Leeds
C0013	Gillard	Simon	Mr	Halifax
C0014	Chan	Tony	Mr	Leeds
C0015	Spencer	Richard	Mr	Bradford
C0016	Beeby	John	Mr	Leeds
C0017	Morley	Mark	Mr	Leeds
C0018	Jones	Alison	Miss	Yeadon
C0019	Cornwell	Richard	Mr	Brighouse
C0020	Crowther	Louise	Mrs	Bradford

Query 1

Select those customers who have both a customer code less than C0010 and who also have the title of “Miss” or customers who live in Halifax.

The SQL for the corresponding selection criteria is:

```
WHERE (CUSTOMER_CODE < 'C0010' AND CUSTOMER_TITLE = 'Miss') OR
      CUSTOMER_TOWN = 'Halifax'
```

To set this criteria, enter the following values in the dialog box. Pay particular attention to the start and end brackets. A tick in the check box next to a bracket indicates that a bracket will be inserted into the SQL.

The output for this query is shown below.

	A	B	C	D	E
1	CUSTOMER_CODE	NAME	FORENAME	TITLE	TOWN
2	C0009	Gorman	Nick	Mr	Halifax
3	C0013	Gillard	Simon	Mr	Halifax
4	C0001	Richardson	Carolyn	Miss	Brighthouse
5	C0004	Welburn	Caroline	Miss	Wakefield
6	C0007	Gothard	Keely	Miss	Leeds
7					

Query 2

Select those customers who have either a customer code less than C0010 or those customers who have the title of “Miss” and who also live in Halifax.

The SQL for the corresponding selection criteria is:

```
WHERE CUSTOMER_CODE < 'C0010' AND
(CUSTOMER_TITLE = 'Miss' OR CUSTOMER_TOWN = 'Halifax')
```

To set this criteria, enter the following values in the dialog box. Pay particular attention to the start and end brackets.

The output for this query is shown below.

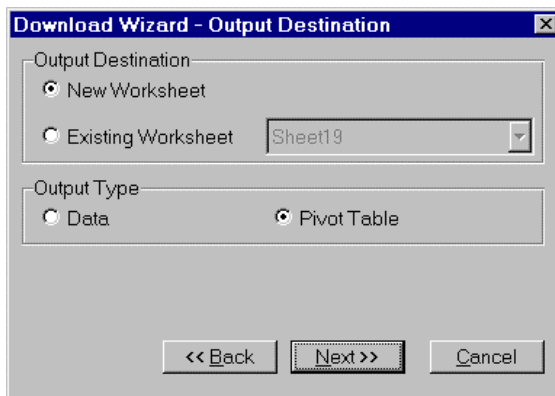
	A	B	C	D	E
1	CUSTOMER_CODE	NAME	FORENAME	TITLE	TOWN
2	C0009	Gorman	Nick	Mr	Halifax
3	C0001	Richardson	Carolyn	Miss	Brighouse
4	C0004	Welburn	Caroline	Miss	Wakefield
5	C0007	Gothard	Keely	Miss	Leeds
6					

Pivot Table

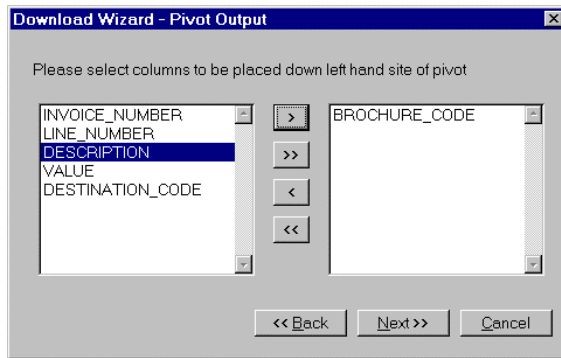
Pivot tables are part of Microsoft Excel. They allow you to view data in many different ways, and you can change the way you view the data with ease. Here you will see how to initially set up a pivot table. Once you have created a pivot table, you can use the facilities that are available in Excel to modify the pivot table.

For exemplary purposes, say that we want to see from which brochures holidays to the various destinations have been purchased.

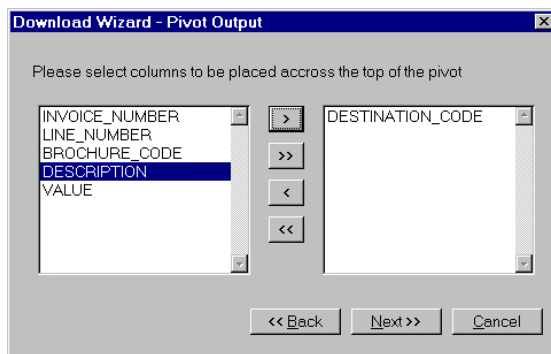
1. Select **Rms, Download Wizard**. You will be presented with the Table Download dialog box. Select the DETAILS table, and then work through the dialog boxes that will follow to select all the columns; do not set any selection criteria and do not sort the data.
2. The Output Destination dialog box appears. Select a new worksheet as the output destination. Send the data to a pivot table by selecting the Pivot Table option. Click the **Next>>** button.



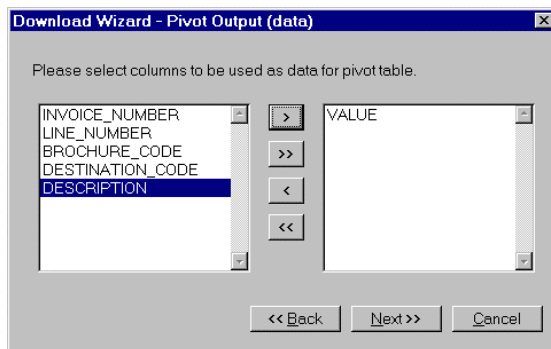
3. The first of the Pivot Output dialog boxes appears. Highlight BROCHURE_CODE and click the **>** button (you cannot select multiple columns). Then click the **Next>>** button.



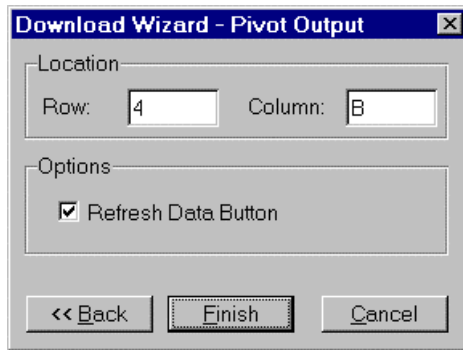
4. The second of the Pivot Output dialog boxes appears. Highlight **DESTINATION_CODE** and click the **>** button (you cannot select multiple columns). Then click the **Next>>** button.



5. The Pivot Output (data) dialog box appears. Highlight **VALUE** and click the **>** button (you cannot select multiple columns). Then click the **Next>>** button.



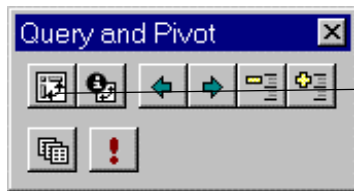
6. The last of the Pivot Output dialog boxes appears. Accept the default options and then click the **Finish** button.



7. The pivot table is created, and you will see something like this

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Refresh														
2															
3															
4		Sum of VALUE	DESTINATION_CODE												
5		PROCEDURE_CODE	D0003	D0004	D0007	D0010	D0012	D0013	D0014	D0015	D0016	D0019	D0022	D0032	D0037
6		E0001	0	0	0	0	0	0	0	0	0	0	0	0	0
7		E0002	0	0	0	0	0	0	0	0	0	0	0	0	0
8		E0003	0	0	0	0	0	0	0	0	1887.9	0	0	0	0
9		E0004	0	0	0	0	0	0	0	0	0	0	5863.2	0	0
10		E0005	0	0	0	0	1911	0	0	0	0	0	0	0	0
11		E0006	0	0	0	0	0	0	0	0	0	0	0	0	0
12		E0011	0	0	0	0	0	0	0	0	0	4191.6	417.3	0	0
13		E0017	0	0	0	0	0	0	0	0	0	0	0	0	0
14		E0019	0	0	0	0	0	0	0	0	0	0	0	0	0
15		E0020	0	0	0	0	0	0	0	0	0	0	0	0	0
16		E0025	0	0	0	0	0	0	0	0	0	0	0	0	0
17		E0029	0	0	0	0	0	0	0	0	0	0	0	0	0
18		E0031	0	0	0	0	0	0	0	0	0	0	0	0	0
19		E0040	0	0	0	0	0	0	0	0	0	0	0	0	0
20		E0041	0	0	0	1087.8	0	0	0	0	0	0	0	0	0
21		E0042	0	0	0	0	0	2091.6	0	0	0	0	0	0	0
22		E0048	0	0	0	0	0	735	0	0	0	0	0	0	0
23		E0049	0	0	0	0	2095.8	0	0	0	0	0	0	0	0
24		E0050	0	0	0	0	0	0	3143.7	3822	0	0	0	0	0
25		E0051	0	1257.9	0	0	0	0	0	0	0	0	0	0	0
26		E0054	1837.5	0	0	0	0	0	0	0	0	0	0	0	0
27		E0057	0	0	0	0	0	0	0	0	0	0	0	1677.9	0
28		E0059	0	0	0	0	0	0	0	0	0	0	0	0	0
29		E0063	0	0	0	0	0	0	0	0	0	0	0	0	0
30		E0067	0	0	0	0	0	0	0	0	0	0	0	0	0
31		E0068	0	0	0	0	0	0	0	0	0	0	0	0	0
32		E0069	0	0	0	0	0	0	0	0	0	0	0	0	0
33		E0070	0	0	0	0	0	0	0	0	0	0	0	0	0
34		E0074	0	0	186.9	0	0	0	0	0	0	0	0	0	0
35		E0075	0	0	291.9	0	0	0	0	0	0	0	0	0	0
36		E0076	0	0	0	0	0	0	0	0	0	0	0	0	2095.8
37		Grand Total	1837.5	1257.9	478.8	1087.8	4006.8	2826.6	3143.7	3822	1887.9	4191.6	6281.1	1677.9	2095.8

8. To modify the pivot table, you will need to use the Excel Pivot Table Wizard. Either ensure that the Query and Pivot toolbar is available (**View, Toolbars...** and then select the Query and Pivot option), or use the **Data, Pivot Table Wizard** menu options.



Pivot Table Wizard on Query and Pivot toolbar.

Upload Data

To upload data, the first row of the worksheet must contain the column names for the table you wish to upload. The data must appear on sequential lines following this. The upload stops at the first blank line. If you have previously used the Download Wizard to obtain data (or even just the column names), then you can be confident that the correct names are being used in a later upload.

After the upload has been performed, the colour of the primary key fields of the uploaded rows will change:

row colour	indicates
red	update or insert failed
blue	row inserted successfully
green	row updated successfully

The rules for updates are as follows:

- First attempt an update (green)
- If the update is unsuccessful, then attempt to insert the row (blue)
- If the attempted insertion is unsuccessful, then indicate failure (red)

Upload Example

This example is for an upload of invoice data. Say that we want to upload two new rows.

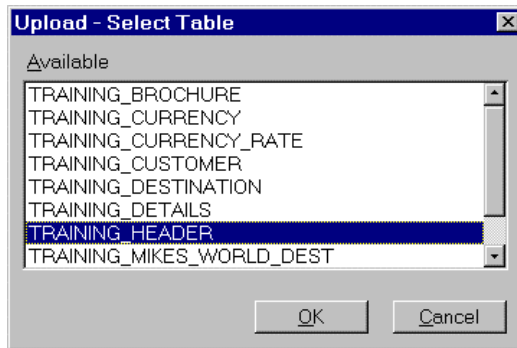
1. Insert the column names into the worksheet. In this example, we did this by downloading some data (for invoice number 20007) and requesting that the column headings be shown.

	A	B	C	D	E	F	G	H	I
1	INVOICE_NUMBER	LINE_NUMBER	INVOICE_TYPE	INVOICE_DATE	INVOICE_VALUE	CODE	CURRENCY		Refresh
2	2007	0	PINV	30-Jun-1997	4,990.00	S0008	GBP		
3									

2. Enter the data for the rows you want to insert. Ensure that there are no blank lines between the rows. In this example, two additional rows were created, shown below. The first new row contains a data type error (INVOICE_VALUE should not contain any non-numeric data).

	A	B	C	D	E	F	G	H	I
1	INVOICE_NUMBER	LINE_NUMBER	INVOICE_TYPE	INVOICE_DATE	INVOICE_VALUE	CODE	CURRENCY		Refresh
2	2007	0	PINV	30-Jun-1997	4,990.00	S0008	GBP		
3	2990	0	PINV	13-Jul-1997	£999.99	S0008	GBP		
4	2991	0	PINV	13-Jul-1997	888.00	S0009	GBP		
5									

3. Start the upload by selecting **Rms, Upload**. The Upload - Select Table dialog box appears.



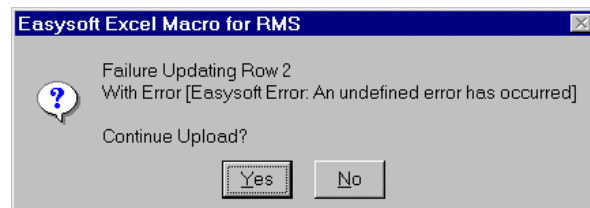
4. Select the table that will be updated and click the **OK** button. In this example, we used the **HEADER** table.
5. The data is uploaded, and the worksheet contains information on the status of the upload.

	A	B	C	D	E	F	G	H	I	J	K	L
1	INVOICE_NUMBER	LINE_NUMBER	INVOICE_TYPE	INVOICE_DATE	INVOICE_VALUE	CODE	CURRENCY		Refresh			
2	2007	0	PINV	30-Jun-1997	4,990.00	S0008	GBP					
3	2990	0	PINV	13-Jul-1997	£999.99	S0008	GBP		Easysoft Error: An undefined error has occurred			
4	2991	0	PINV	13-Jul-1997	888.00	S0009	GBP					
5												

After the update was performed, the colours on the worksheet changed. The key fields on the first row is green, indicating successful update (although no data values were modified). The second row is red, indicating that the attempted update was not successful and an error message is presented.

Note - Auto Skip Failed Updates

If the Auto Skip Failed Updates option is not selected in the Options dialog box (page 12-5), then each time that an error occurs, a message will be presented asking whether you wish to continue the update.



Report

This option allows you to create reports tailored to your exact needs. There are two options, **Setup New Report**, which is used to define the report, and **Run Report**.



Setup New Report

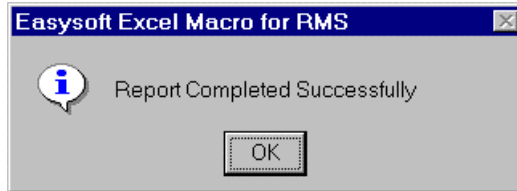
1	Report Name	Author						
2	simple	Mike						
3								
4	Worksheet Row	Column	Headings	Format	SQL	Last Rows	Last Time	
5	Sheet5	1	1	YES	YES	SELECT * FROM TRAINING_BROCHURE WHERE BROCHURE_CODE <'B0010'	9	1seconds
6								

1. Enter values for **Report Name** and **Author** at the top of the report. These are optional, and do not affect the output.
2. **Worksheet Row**. This specifies the location of the output. It must be a worksheet that currently exists, and it must not be the worksheet on which the Setup Report resides. Remember to enter the name of the worksheet exactly as it appears on the tabs at the bottom of the screen.
3. **Row** and **Column** specify the first row and column respectively on which the data is to be returned.
4. **Headings**. If **YES** is specified, then column headings will be included in the report. This option may be left blank (so that headings are not shown).
5. **Format**. If **YES** is specified, then the width of each column is adjusted to fit the data returned. This option may be left blank (i.e. no formatting).
6. **SQL**. This is the SQL that is sent to the server. Multiple rows of SQL are allowed (see “Split SQL Lines”, page 12-19).
7. Save the workbook if you want to use the report in the future. The report is now ready to run.

Note: **Last Rows** and **Last Time** are filled in automatically when the report is run. They specify the number of rows returned and the finishing time of the report.

Run Report

1. Select the worksheet that contains the report you wish to run.
2. Run the report (**Rms, Report, Run Report**). You should see a message indicating success. Click **OK**.



3. Switch to the worksheet on which you asked for the data to be placed. Example output:

	A	B	C
1	BROCHURE_CODE	SUPPLIER_CODE	DESCRIPTION
2	B0001	S0001	Faraway Shores
3	B0002	S0001	Ski-ing
4	B0003	S0001	Young At Heart
5	B0004	S0001	Price Breakers
6	B0005	S0001	City Breaks
7	B0006	S0001	World Wide
8	B0007	S0001	Flight Only
9	B0008	S0001	Ala Carte

Advanced Techniques

This section outlines a few techniques you can use to improve the reports you write.

Split SQL Lines

The SQL that you write need not be entered on just one line. To improve readability, you can enter it over many lines, as shown in the screen shot below.

	A	B	C	D	E	
1	Report Name	Author				
2	Split	Mike				
3						
4	Worksheet	Row	Column	Headings	Format	SQL
5	Sheet1	1	1	YES	YES	SELECT *
6						FROM TRAINING_SUPPLIER
7						WHERE SUPPLIER_CODE = 'S0004'
8						

You must not have any empty lines between the lines of SQL - the first empty line is taken to be the end of the query.

You cannot split words over lines, thus the following text would result in an error message.

SQL	
SELECT *	
FROM TRAINING_SUPPL	
LIER WHERE SUPPLIER_CODE = 'S0004'	

Create Generic Report

To make a generic report, you can use cell references. For example, say you regularly ran reports to determine which brochures were obtained from a given supplier, you could create the report such that the supplier code is entered in a prominent place on the report; using cell references, this would then be entered into the SQL query.

Say we want this SQL query: `SELECT * FROM TRAINING_BROCHURE WHERE SUPPLIER_CODE='<supplier code>'`

When the supplier code is changed, the value will change in the SQL.

In this example, to aid clarity in the screen shots, we will also split the SQL over three lines. The first two lines are not relevant to the exercise, but are needed for the query in its entirety.

1. Enter **supplier code** in cell E1 - this will be the heading under which to enter the supplier code.
2. Enter the value of the supplier code in cell E2. In the example, we use S0004. Although the code is a character string, the quote marks needed for this are not necessary here, as they will be included in the SQL text that is input later.
3. Select the cell in which you will write first line of the SQL (F6). Enter the following:
`SELECT * FROM TRAINING_SUPPLIER`
4. Select the cell in which you will write second line of the SQL (F7).
5. Move the cursor to the Formula bar.

	C	D	E	
1	Author		Supplier Code	
2	mike		S0004	
3				
4	Column	Headings	Format	SQL
5	1	YES	YES	
6				

6. Press the equals (=) key (this indicates that a formula is to follow).

7. Type: `" WHERE SUPPLIER_CODE = ' "`

Ensure you type the quote marks exactly as shown. After the first quote mark there is a space character. The last three characters of the text are the equal sign, single quote, double quote. Notice how the text also appears in the SQL cell (see next screen shot).

8. Press the space bar.
9. Press the ampersand (&) key.
10. Press the space bar.
11. Select the cell (E2) containing the supplier code.
12. Notice how the value E2 is added to the formula (you could have typed it directly).

The screenshot shows an Excel spreadsheet with the following data:

	C	D	E
1	Author		Supplier Code
2	Mike		S0004
3			
4	Column	Headings	Format SQL
5	1	YES	YES SELECT * FROM TRAINING_SUPPLIER
6			" WHERE SUPPLIER_CODE = " & E2
7			

The formula bar at the top shows the formula in cell F6: `= " WHERE SUPPLIER_CODE = " & E2`

13. Press the following sequence of characters:
space bar, ampersand (&) key, space bar, double quote, single quote, double quote.
14. Press the Enter key. The SQL is ready to run.
15. Notice how the formula contains a reference to cell E2 which contains the company name. If you change the name, then this change will automatically be reflected in the SQL.

Troubleshooting

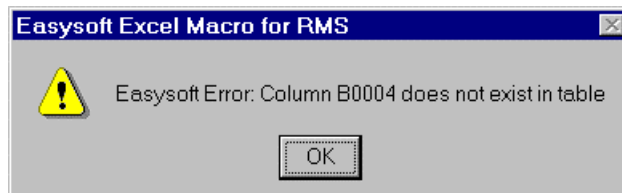
Error: Column x does not exist

This occurs when you write SQL in a custom report if the following conditions apply:

- the query contains a condition of the form WHERE <column> <operator> <value>
- <value> is a character datatype
- the data that you entered for <value> was not enclosed within character literals, that is, single quote marks.

For example if you wrote: WHERE SUPPLIER_CODE < B0004
instead of WHERE SUPPLIER_CODE < 'B0004'

Then the following error message would occur:



The reason for this is that the syntax of the first where clause is correct; it is possible to compare values from two columns. Since there is no column called B0004 the query returns the error message.

There are two (or more) Refresh buttons on the worksheet.

	A	B	C	D	E	F	G
1	CUSTOMER_CODE	NAME	FORENAME	TITLE	TOWN	Refresh	Refresh
2	C0009	Gorman	Nick	Mr	Halifax		
3	C0013	Gillard	Simon	Mr	Halifax		
4	C0001	Richardson	Carolyn	Miss	Brighouse		
5	C0004	Welburn	Caroline	Miss	Wakefield		
6	C0007	Gothard	Keely	Miss	Leeds		
7	C0006	Berlin	Karl	Mr			
8	C0007	Gothard	Keely	Miss			
9	C0008	Naylor	Ian	Mr			

In the general case, this is not a problem; each different query has an associated Refresh button, and provided that the column headings or the result sets do not overlap, no confusion arises.

However, in the example above, the Refresh button in column F belongs to one query which uses the first four columns. The Refresh button in column G belongs to a different query which uses the first five columns.

If you want to have more than one query on the same worksheet, then ensure you plan the layout carefully. If you want to clear a worksheet of a button, then select the area to delete, and use **Edit, Delete** from the main menu (**Clear** removes data, but it will not remove a button).

Appendices

Contents

Appendix A. Global Glossary	3
Appendix B. Data Types	10
SQL Data Types	10
Local Data Types	11
Appendix C. Import Export CSV Format	15
Appendix D. Easysoft ODBC PC Installation	16
Starting the Driver Installation	16
Appendix E. Easysoft COBOL Converter	20
Installation	20
Using the Converter	21
Basic Conversion	22
Output FILLER Fields	24
Output Group Items (sub-structured fields)	24
Convert Raw Record Descriptions	25
Notes / Information	26
COBOL Data Types	27
Appendix F. Easysoft PowerHouse PDL Converter	30
Using the Converter	30
Basic Conversion	32
Output FILLER Fields	33
Output Sub-structured Fields	34
Stripping Characters	35
Define Conditions for use by the Pre-processor	35
Notes / Information	36
Appendix G. Troubleshooting	37
Easysoft ODBC Logging	38
EASYSOFT.INI	39
Microsoft Trace Options	41
Version 3.0 Trace Options	41
SQL.LOG Described	42
How to Work Out What the Server is Doing	43
Find the Process Causing the Problem	43
View Log File	45
Stop the Process	45
Appendix H. Using Easysoft Support Services	47
Support Check List	48
The Easysoft FTP Site	49
Sending Data to Easysoft	50

Appendix I. RMS in Context	51
Overview of OpenVMS	53
Digital Command Language	56
Global Index	A-60

APPENDIX A

Global Glossary

\$ DCL prompt.

! DCL comment separator.

Access Control List see ACL.

ACL (Access Control List) A list that defines the kinds of access to be granted or denied to users of an object.

Alpha A type of computer produced by Digital Equipment Corporation (DEC) which can use the VMS and OpenVMS operating systems.

Alternate index An index which is additional to the primary index.

Alternate key (in RMS terms) An optional key in an indexed file.

ANSI (American National Standards Institute) The primary formal standards-making body in the United States.

API (Application Programming Interface). A set of related functions that a computer programmer uses to obtain some kind of service from another piece of software.

Application (In ODBC terms) a program that processes data and which runs under Microsoft Windows.

Area (in RMS terms) A region of an RMS indexed file that allows a user to specify placement and/or specific bucket sizes for particular portions of a file. An area consists of any number of buckets; there may be from 1 to 255 areas in a file.

ASCII (American Standard Code for Information Exchange) A set of 8-bit binary numbers which represent the alphabet, punctuation, numerals and other special symbols which are used to represent text.

Bit A binary digit (0 or 1). The smallest unit of information in a binary notation system.

Block (In general terms) the smallest unit of space into which a mass storage device can be divided.

(In OpenVMS terms) The smallest logically addressable unit of data that a device can transfer in an I/O (Input/Output) operation. Typically, 512 contiguous bytes.

Block I/O A data accessing technique in which the program manipulates the blocks (physical records) that compose a file, rather than its logical records. This allows for the direct access of the blocks in a file without regard for the file organisation or record format.

Blocked record (in VAX/OpenVMS terms) A record in a file in which more than one record or segment can be contained in a block.

Boolean Boolean logic is the two-valued logic of true and false.

Bucket (in VAX/OpenVMS terms) A storage structure of between 1 and 32 disk blocks that is used to store and transfer units of data in files which have a relative or indexed file organisation. A bucket can contain only complete records, unlike a block.

Buffer A temporary storage space for data.

Byte A binary character string consisting of bits operated on as a unit.

Catalog (In SQL terms) a named collection of one or more tables grouped together. The catalog contains definitions that describe SQL features of application databases, such as which columns belong to which tables, user privileges, etc.

CLI (Call Level Interface) A set of function calls that a computer programmer uses to obtain some kind of service from another piece of software. CLI is used with reference to SQL standards to describe an interface that is not embedded SQL.

Column the vertical dimension of a table (cf. row). The field at the intersection of a row and a column holds the value in accordance with the data type specified.

Comma Separated Text see CSV.

Command An instruction (usually an English word) that specifies an operation to be performed.

Conformance level (API) Refers to the set of ODBC functions that an ODBC driver supports.

Conformance level (SQL) Refers to the set of SQL functions that an ODBC driver supports.

Contiguous A technical term meaning adjacent, next to or continuous.

CST (Comma Separated Text). see CSV.

CSV (Comma Separated Values). CSV is text that defines data values. The values are separated by commas (same as CST).

Data source A set of database files plus (if appropriate) the associated operating system, DBMS and network.

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

Database A collection of data files.

DBMS (Database Management System). Software that handles access to a database.

DECNET A proprietary communication protocol produced by Digital Equipment Corporation (DEC).

Default A value automatically inserted into a table in the case where a user has not specified a value to be used.

DCL Digital Command Language. The standard common interface to Digital's major operating systems.

DCL prompt By default this is the dollar sign (\$). It indicates that DCL is ready to accept a command.

DLL (Dynamic Link Library). A shared library of code that is loaded in and out of memory as and when it is needed.

Download(ing) In non-technical terms downloading is taking data from the Server and storing it on the Client PC (cf. uploading).

Driver See ODBC driver.

Driver Manager A dynamic link library (DLL) provided by Microsoft, the main function of which is to load ODBC drivers.

FDL (File Definition Language) A language used to describe file organisations of data files.

Field A segment of a data record.

File Definition A file definition in the Easysoft Administrator contains information relating to the organisation and length of a file stored on the Server platform.

File Definition Language see FDL.

File organisation (in VAX/OpenVMS terms) The file structure that is used as the physical arrangement of the data on the storage medium. RMS file organisations are sequential, indexed and relative.

File sharing (in VAX/OpenVMS terms) The capability of a relative or indexed file to allow access to more than one process.

File specification A unique identification for a file which gives its physical location, the file type and the version number.

File type The part of a file specification that consists of a period (.) followed by a number of characters. Same as file name extension.

Floating (1) A number that may be positive or negative but that has a whole (integer) portion and a fractional (decimal) portion. (2) An arithmetic operation in which the decimal point is not fixed, but placed automatically in a correct position.

FTP (File Transfer Protocol). A standard method (rfc 959) of transferring files between different machines.

Function (SQL) A function in SQL takes a *scalar value* or a set of scalar values and returns a scalar value.

Image (in VAX/OpenVMS terms) Procedures and data bound together by the linker. There are three types of image: executable, shareable and system

Index (in DEC terms) The structure that allows retrieval of records in an indexed file by key value. See key.

Indexed Sequential Access Method. See ISAM.

Indexed file organisation (in VAX/OpenVMS terms) A file organisation in which a file contains records and a primary key index (and possibly other alternate key indexes) which is used to process the records sequentially by index or randomly by index.

Interoperability An application that is interoperable is one that can access many different databases.

ISAM (in VAX/OpenVMS terms) (indexed sequential access method) A method of file organisation, storage and access. Records are stored and accessed on the basis of the contents of a field called a key field. An index is used to determine a position from which a short sequential search is made for the desired entry.

ISO (International Standards Organisation) A world-wide federation of national standards bodies that sets standards for a variety of technologies.

Key (in DEC terms) A string or numeric data that specifies a particular record that is accessed randomly. In indexed files the user defines the length and location within the records. RMS uses the key to build an index. In relative files, key refers to the relative record number of each data record in the data file. RMS uses the relative record number to identify and access data records.

Literal A literal is a way of representing a value. Each value has a *data type*, and for each data type there is a corresponding literal specification.

Local Data type A local data type represents the storage format of the data within the data files on the Server. This local data type is mapped onto an SQL data type within the Easysoft SQL Engine.

Logical see Logical name.

Logical name In the OpenVMS and VMS operating systems, an alias for a file specification (can include directories, subdirectories, input/output devices). It is a named variable which is replaced by a value when it is used.

Multi-tier driver (also see single-tier driver) In a multi-tier configuration the ODBC driver sends SQL requests to a server that processes those requests. Typically, multiple-tier systems are divided across platforms.

Multiprogramming Parallel processing of more than one routine or program by interleaving them and timesharing the computer system.

Nibble Four contiguous bits of memory; one half of a byte.

ODBC (Open DataBase Connectivity). An industry standard defined by Microsoft which is by software that allows communication between different database systems.

ODBC driver The software that implements ODBC function calls. Each driver is specific to an application.

OpenVMS An operating system that runs on Alpha and VAX machines (see VMS).

Operating system (1) An integrated collection of programs that controls the execution of computer programs and performs system functions. (2) Software that organises a central processor and peripheral devices into an active unit for the development and execution of programs.

Owner UIC The UIC of the owner of the file. Typically, the person who created the file.

Packed decimal (in VAX/OpenVMS terms) A method of representing a decimal number by storing a pair of decimal digits in 1 byte. This can be done because there are 8 bits in 1 byte and only 4 bits are needed to represent any one of the numbers 0 through to 9.

Packet In communications technology, a packet is the smallest unit of information that can be transmitted over a network.

Parameter A value that is passed to a command.

Parse To break a string of characters into primary components for the purpose of interpreting the string.

Predicate A predicate is a simple statement which can be true, false or sometimes unknown.

Primary key (in RMS terms) The mandatory key within the records of an indexed file.

Privilege A privilege is the right to perform a particular action (e.g. INSERT) on an object (e.g. table).

Protection code A code in each file that indicates who may and may not access the file.

Qualifier A portion of a command string that modifies a command verb or a command parameter. It has the format: /qualifier[=option].

Random access An access mode in which the value of a data item identifies the record.

Record A collection of related data items treated as a unit. A record contains one or more fields.

Record access mode The method used in RMS for retrieving and storing records in a file.

Record locking The ability to control the operations being performed on a file that is being simultaneously accessed by more than one program. It prevents two programs accessing the same record.

Relative file organisation (in VAX/OpenVMS terms) The arrangement of records in a file in which each record occupies a cell of equal length.

Relative record number An identification number that specifies the position of a record cell relative to the beginning of the file.

RMS (Record Management Services) A DIGITAL-specific term. A set of routines that is used to manipulate data files. The OpenVMS high-level file system.

Routine A set of computer instructions that perform an operation.

Row A row is the horizontal dimension of a table (cf. column). A row in its most basic roll equates to a record within a file.

Scalar value A scalar value is in principle any value that can be assigned to a column.

Search condition A search condition is a statement or group of statements joined together by boolean operators which results in a true, false or unknown condition.

Segmented key A key that consists of separate segments (sections) in different parts of a record.

Sequential access An access mode in which records are obtained from a file in a consecutive sequence.

Sequential file organisation (in VAX/OpenVMS terms) A file organisation in which records appear in the order in which they were originally written. The records can be fixed length or variable length.

Server A server can be considered as an engine providing some service. A client calls the server, resulting in the service being performed.

Set function (SQL) An operation on a set of values in a column of a table or all values from a column in a group of rows in a table.

Simple key (RMS term) A key that consists of a single segment (section) of a record.

Single-tier driver (also see multi-tier driver) An ODBC implementation in which the data is processed directly by the ODBC driver.

SQL (Structured Query Language). A standard language for interacting with relational database systems.

SQL statement SQL statements can be categorised as data manipulation, data definition or SQL control statements.

Symbol (OpenVMS) A name that represents numeric, character, or logical values. When you use a symbol in a DCL command line, DCL uses the value you assign to the symbol. By defining a symbol as a command line, you can execute the command by typing only the symbol name.

Syntax The structure of a language, command or statement.

System data source A data source which can be accessed by any user on a given computer cf. user data source.

Table A table consists of column definitions and is the view a user sees of data definitions they have defined. You can consider a table as a rectangular sheet containing *columns* and *rows*. Each intersection (cell) can contain a value.

TCP/IP (Transmission Control Protocol/Internet Protocol). A standard method (rfc 793) of accessing data on different machines.

UIC (in VAX/OpenVMS terms) The abbreviation for user identification code. A 32 bit value assigned to users and files that specifies the allowed operations. A unique identifier for a user in the system.

Upload(ing) In non-technical terms uploading is taking data from the Client PC and storing it on the Server (cf. downloading).

User data source A data source which can be accessed by a specific user only cf. system data source.

User Identification Code see UIC.

VAX (Virtual Address eXtension). A type of computer produced by Digital Equipment Corporation (DEC) which can use the VMS and OpenVMS operating systems.

Virtual memory (in VAX/OpenVMS terms) The set of storage location in physical memory and on disk that is referred to by virtual addresses. From a programming viewpoint, the secondary storage locations appear to be locations in physical memory.

VMS (Virtual Management System). An operating system that runs on VAX and Alpha machines. Superseded by OpenVMS.

Volume (in VAX/OpenVMS terms) A mass storage medium such as a disk pack.

X/Open SQL Access Group An industry consortium of DBMS vendors whose objective is to enable SQL-based products from multiple vendors to work together.

APPENDIX B

Data Types

This appendix consists of 2 sections:

- SQL data types that are supported by Easysoft ODBC
- Local data types supported by the Easysoft SQL Engine on the Server

SQL Data Types

This section lists the SQL data types that are supported by Easysoft ODBC.

Character String Data Types

CHAR Character string of fixed string length n , where n is less than or equal to 254.

VARCHAR Variable length character string with a maximum string length 254.

Exact Numeric Data Types

TINYINT Signed, exact, numeric value with precision 3 and scale 0.

SMALLINT Signed, exact, numeric value with a precision 5 and a scale 0.

INTEGER Signed, exact, numeric value with a precision 10 and a scale 0.

Approximate Numeric Data Types

DOUBLE Signed, approximate, numeric value with a mantissa precision 15 (zero or absolute value 10^{-38} to 10^{38}).

Datetime Data Types

DATE Date data.

TIME Time data.

TIMESTAMP Date/time data.

Local Data Types

The supported data types of the Easysoft SQL Engine on the Server are shown in the table below.

Notes

1. Column headings:

Storage type. The options are: A = ASCII, B = Binary, P = Packed

Length: If length is *not* fixed, then the default length is shown first, and the maximum length is shown in brackets.

Format (for dates and times): Y = Year, M = Month, D = day, h = hour, m = minute, s = second, c = 0.01 second.

Reversed: an R in this column indicates that the byte order is reversed.

Un/Signed: S indicates that the data type is signed, U indicates that it is not signed.

2. (d) after an entry indicates that it is a default value and can be changed.

3. For historical reasons, data types with different names may have the same structure.

4. Additional notes (indicated by a number after the data type name) on a few datatypes are contained in the table on page 14.

Local Data Types

Name	Storage type	Default SQL data type	Length	Precision	Scale	Format	Start date/time	Interval	Reversed	Un/Signed
ABC-PACKED (1)	P	DOUBLE	8 (16)	16 (d)	0 (d)	Trailing sign	-	-	-	S
BASIC-DATE	B	DATE	2	5	0	DDDDYY	01-JAN-1970	1 day	-	S
BASIC-DATE4	B	DATE	4	10	0	DDDDYY	01-JAN-1970	1 day	-	S
BINDATE	B	DATE	4	10	0	YYYYMMDD (d)	-	-	-	S
BINTIME	B	TIME	4	10	0	hhmmsscc (d)	-	-	-	S
BINTIME-U	B	TIME	4	10	0	hhmmsscc (d)	-	-	-	U
BINTIME-2-U	B	TIME	2	5	0	hhmm (d)	-	-	-	U
BYTE	B	TINYINT	1	3	0	-	-	-	-	S
C-TIME (2)	B	TIMESTAMP	4	10	0	-	01-JAN-1970 00:00:00.00	1 sec.	-	S
C-TIME-REV (3)	B	TIMESTAMP	4	10	0	-	01-JAN-1970 00:00:00.00	1 sec.	R	S
C-TIME2 (4)	B	TIMESTAMP	4	10	0	-	01-JAN-1900 00:00:00.00	1 sec.	-	S
CDS-DATE (5)	B	DATE	2	5	0	-	01-JAN-1900	1 day	-	S
CDS-DATE2 (6)	B	DATE	2	5	0	-	24-OCT-1924	1 day	R	S
CDS-TIME3 (7)	B	TIME	3	9	0	hhmmss	-	-	-	S
CODA-DATE2 (8)	B	TIMESTAMP	4	10	0	-	18-NOV-1858 00:00:00.00	1 minute	-	S
D-FLOATING	B	DOUBLE	8	15	0 (d)	-	-	-	-	S

Table continued on next page

Name	Storage type	Default SQL data type	Length	Precision	Scale	Format	Start date/time	Interval	Reversed	Un/ Signed
DATE	A	DATE	11 (254)	11 (d)	0	DD-MMM-YYYY	-	-	-	-
DIBOL-OVERPUNCHED-NUMERIC	A	DOUBLE	16 (19)	16 (d)	0 (d)	Overpunched trailing sign	-	-	-	S
DOUBLE	B	DOUBLE	8	15	0 (d)	-	-	-	-	S
F-FLOATING	B	DOUBLE	4	7	0 (d)	-	-	-	-	S
FLOAT	B	DOUBLE	4	7	0 (d)	-	-	-	-	S
FREEFORM	A	VARCHAR	16 (32)	16 (d)	0 (d)	-	-	-	-	-
G-FLOATING	B	DOUBLE	8	15	0 (d)	-	-	-	-	S
H-FLOATING	B	DOUBLE	16	33	0 (d)	-	-	-	-	S
IBASEDATE (9)	B	TIMESTAMP	8	19	0	-	17-NOV-1858 00:00:00.00	1×10 ⁻⁷ sec.	-	S
INFORMIX-DATE-4	B	DATE	4	10	0	-	31-DEC1899	1 day	-	S
INFORMIX-DATE-4-REV	B	DATE	4	10	0	-	31-DEC1899	1 day	R	S
INTEGER1	B	SMALLINT	1	3	0 (d)	-	-	-	-	S
INTEGER2	B	SMALLINT	2	5	0 (d)	-	-	-	-	S
INTEGER3	B	INTEGER	3	7	0 (d)	-	-	-	-	S
INTEGER4	B	INTEGER	4	10	0 (d)	-	-	-	-	S
INTEGER5	B	DOUBLE	5	12	0 (d)	-	-	-	-	S
INTEGER6	B	DOUBLE	6	15	0 (d)	-	-	-	-	S
INTEGER7	B	DOUBLE	7	17	0 (d)	-	-	-	-	S
INTEGER8	B	DOUBLE	8	19	0 (d)	-	-	-	-	S
INTEGER-S	B	INTEGER	4	10	0 (d)	-	-	-	-	S
JDATE	B	DATE	2	5	0	-	01-JAN-1900	-	-	S
KD-DATE (10)	A	DATE	7	7	0	YYYYDDD	-	-	-	-
LEADING-OVERPUNCHED-NUMERIC	A	VARCHAR	16 (32)	16 (d)	0 (d)	Overpunched leading sign	-	-	-	S
LEADING-SIGN-NUMERIC	A	VARCHAR	16 (32)	16 (d)	0 (d)	Leading sign	-	-	-	S
LONG	B	INTEGER	4	10	0 (d)	-	-	-	-	S
LONGWORD	B	INTEGER	4	10	0 (d)	-	-	-	-	S
LONGWORD-REVERSED	B	INTEGER	4	10	0 (d)	-	-	-	R	S
NUMERIC	A	VARCHAR	16 (32)	16 (d)	0 (d)	-	-	-	-	-
PACKED	P	DOUBLE	8 (16)	16 (d)	0 (d)	Trailing sign	-	-	-	S
PACKED-S	P	DOUBLE	8 (16)	16 (d)	0 (d)	Trailing sign	-	-	-	S
PACKED-U	P	DOUBLE	8 (16)	16 (d)	0 (d)	-	-	-	-	U
PHDATE (11)	B	DATE	2	5	0	-	-	-	-	S
PHDATE4 (12)	B	DATE	4	10	0	-	-	-	-	S
PHDATETIME	B	TIMESTAMP	8	19	0	YYYYMMDD hhmssc	-	-	-	U
PSI-DATE (13)	P	DATE	4	7	0	YYYYDDD	-	-	-	S
QUADWORD	B	DOUBLE	8	19	0 (d)	-	-	-	-	S
QUADWORD-REVERSED	B	DOUBLE	8	19	0 (d)	-	-	-	R	S
SEB-DATE	B	DATE	6	15	0	YYYYMMDD	-	-	-	S
SINT3	B	INTEGER	3	7	0 (d)	-	-	-	-	S
SINT5	B	DOUBLE	5	12	0 (d)	-	-	-	-	S
SINT6	B	DOUBLE	6	15	0 (d)	-	-	-	-	S

Table continued on next page

Name	Storage type	Default SQL data type	Length	Precision	Scale	Format	Start date/time	Interval	Reversed	Un/ Signed
SINT7	B	DOUBLE	7	17	0 (d)	-	-	-	-	S
SMITHSONIAN-DATE (14)	B	DATE	4	10	0	-	17-NOV-1825	1 day	-	S
SPD-DATE (15)	P	DATE	16	15	0	YYYYMMDD (d) Trailing sign	-	-	-	S
SPECIAL-DATE-1	B	DATE	2	5	0	-	05-AUG-1948	1 day	-	S
SPECIAL-DATE-2	B	DATE	2	5	0	-	31-DEC-1919	1 day	-	U
SPECIAL-DATE-3	N	DATE	5	5	0	-	30-DEC-1899	1 day	-	S
SPECIAL-DATE-4	A	DATE	5 (5)	5	0	YYDDD	-	-	-	-
STARDATE (16)	B	TIMESTAMP	8	20	0	-	17-NOV-1858	1×10 ⁻⁷ sec.	-	S
STRING	A	VARCHAR	50 (254)	50 (d)	0	-	-	-	-	-
TIME	A	TIME	11 (254)	11 (d)	0	hh:mm:ss.cc (d)	-	-	-	-
TIMESTAMP (17)	B	TIMESTAMP	8	20	0	-	17-NOV-1858	1×10 ⁻⁷ sec.	-	S
TRAILING-OVERPUNCHED-NUMERIC	A	VARCHAR	16 (32)	16 (d)	0 (d)	Overpunched trailing sign	-	-	-	S
TRAILING-SIGN-NUMERIC	A	VARCHAR	16 (32)	16 (d)	0 (d)	trailing sign	-	-	-	S
UINT3	B	INTEGER	3	8	0 (d)	-	-	-	-	U
UINT5	B	DOUBLE	5	13	0 (d)	-	-	-	-	U
UINT6	B	DOUBLE	6	15	0 (d)	-	-	-	-	U
UINT7	B	DOUBLE	7	17	0 (d)	-	-	-	-	U
UINTEGER1	B	SMALLINT	1	3	0 (d)	-	-	-	-	U
UINTEGER2	B	INTEGER	2	5	0 (d)	-	-	-	-	U
UINTEGER3	B	INTEGER	3	8	0 (d)	-	-	-	-	U
ULONG	B	DOUBLE	4	10	0 (d)	-	-	-	-	U
ULONGWORD-REVERSED	B	DOUBLE	4	10	0 (d)	-	-	-	R	U
UQUADWORD	B	DOUBLE	8	20	0 (d)	-	-	-	-	U
UQUADWORD-REVERSED	B	DOUBLE	8	20	0 (d)	-	-	-	R	U
UWORD	B	INTEGER	2	5	0 (d)	-	-	-	-	U
UWORD-REVERSED	B	INTEGER	2	5	0 (d)	-	-	-	-	U
VMSDATE4 (18)	B	DATE	4	10	0	-	17-NOV-1858	1×10 ⁻⁷ sec.	-	S
VMSDATE4-REVERSED (19)	B	DATE	4	10	0	-	17-NOV-1858	1×10 ⁻⁷ sec.	R	S
VMSDATE8 (20)	B	DATE	8	20	0	-	17-NOV-1858	1×10 ⁻⁷ sec.	-	S
VMSDATE8-REVERSED (21)	B	DATE	8	20	0	-	17-NOV-1858	1×10 ⁻⁷ sec.	R	S
WORD	B	SMALLINT	2	5	0 (d)	-	-	-	-	S
WORD-REVERSED	B	SMALLINT	2	5	0 (d)	-	-	-	-	S
ZONED	A	VARCHAR	16 (32)	16 (d)	0 (d)	-	-	-	-	U
ZONED-NUM	A	VARCHAR	16 (32)	16 (d)	0 (d)	Trailing sign	-	-	-	S
ZONED-S	A	VARCHAR	16 (32)	16 (d)	0 (d)	Trailing sign	-	-	-	S
ZONED-U	A	VARCHAR	16 (32)	16 (d)	0 (d)	-	-	-	-	U

Notes on Local Data Types		
Note number	Datatype	Comment
1	ABC-PACKED	Sign is stored in last nibble (4 bits) of packed string. Positive sign represented by 3, negative sign represented by 5
2	C-TIME	Count of seconds since Start
3	C-TIME-REV	Count of seconds since Start
4	C-TIME2	Count of seconds since Start
5	CDS-DATE	Each year has 366 days
6	CDS-DATE2	Each year has 366 days
7	CDS-TIME3	CDS format
8	CODA-DATE2	Count of minutes since Start
9	IBASEDATE	Begin date: 01-JAN-0100 00:00:00.00, End date: 09-DEC-5941 00:00:00.00
10	KD-DATE	DDD is number of days since start of year. Example: 20th February 1996 is stored as 1996051 (051= 31 days Jan, + 20 days Feb).
11	PHDATE	2 byte storage, first 7 bits for year, next four for month, last five for day.
12	PHDATE4	4 byte storage, first 23 bits for year, next four for month, last five for day.
13	PSI-DATE	DDD is number of days since start of year. Example: 20th February 1996 is stored as 1996051 (051= 31 days Jan, + 20 days Feb).
14	SMITHSONIAN-DATE	Count of days since Start
15	SPD-DATE	Signed packed decimal date
16	STARDATE	End date: 09-dec-5941
17	TIMESTAMP	End date: 31-DEC-9999
18	VMSDATE4	End date: 31-DEC-9999
19	VMSDATE4-REVERSED	End date: 31-DEC-9999
20	VMSDATE8	End date: 31-DEC-9999
21	VMSDATE8-REVERSED	End date: 31-DEC-9999

APPENDIX C

Import Export CSV Format

This appendix lists the import/export CSV format used by the Easysoft PC Administrator and the Host Administrator on the server.

Angle brackets (<>) indicate that some value is substituted for the text shown inside the braces; the brackets are not part of the file definition.

File Definition

"FILE", "<File Name>", "<File Organisation>", "<Record Type>", "<Record Size>"

Field Definition

"FIELD", "<File Name>", "<Field Name>", "<Data Type>", "<Offset>", "<Length>", "<Precision>", "<Scale>", "<Encrypted>", "*" "<Date Format>", "<Default>"

or

"FIELD", "<File Name>", "<Field Name>", "<Data Type>", "<Offset>", "<Length>"
(Other details are defaulted by the import routine).

* indicates only used for ASCII Dates/Times.

Database Definition

"DB", "<Database Name>", "<Default Directory>", "<Driver Name>",
"<Connect String>"

Table Definition

"TABLE", "<Database Name>", "<Table Name>", "<Table Type>", "<File Name>",
"<File Specification>", "<Criteria>"

Column Definition

"COLUMN", "<Database Name>", "<Table Name>", "<Column Name>",
"<Field Name>", "<SQL Data Type>", "<Length>", "<Updatable>", "<Visible>", "<Default Value>"

User Definition

"USER", "<User Name>", "<Password>"

Note: The password field is not written on exports

Database Privileges Definition

"DBPRV", "<User Name>", "<Database Name>", "<Database Username>", "<Database Password>", "<Allow Select>", "<Allow Insert>", "<Allow Update>", "<Allow Delete>",
"<Security Level>"

Table Privileges Definition

"TBLPRV", "<Database Username>", "<Database Name>", "<Database Password>",
"<Table Name>", "<Allow Select>", "<Allow Insert>", "<Allow Update>", "<Allow Delete>"

APPENDIX D

Easysoft ODBC PC Installation

This chapter describes the installation of the Easysoft ODBC Driver.

Easysoft ODBC requires Windows 95 or above and unless there are differences which need to be considered, these platforms are collectively referred to as “Windows”.

Easysoft ODBC storage requirement: 5 megabytes

Easysoft ODBC memory requirement: 4 megabytes

To upgrade from a previous Easysoft version, follow the procedure outlined for installing the software.

Note: When you upgrade the ODBC driver, you must also upgrade the Easysoft Server Component on the server and the version number of these two products must be identical.

Starting the Driver Installation

There are three ways to obtain the Easysoft ODBC Driver software:

- 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-RMS Driver section of the web site. Log in. (If you have not yet done so, you need to register first. On the registration form, an asterisk (*) indicates that a field is mandatory.) From the download page, choose the client platform release that you require.
- 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. Change to the pub/download/client/ directory and then choose the platform release that you require.
- You can order Easysoft software on CD by email, telephone or post (see Contacting Easysoft).

The name of the Easysoft ODBC Driver distribution file is of the form:

- `odbc-x.y.z-windows-x86.exe`

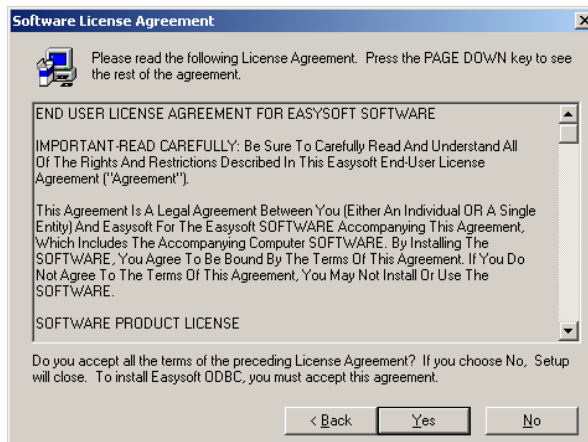
where "x" is the major version number, "y" is the minor version number and "z" is the build index.

Driver Installation Steps

Execute the file distribution that you obtained from one of the sources described earlier in this chapter. There will be a short delay while Setup prepares the wizard to guide you through the installation procedure. Then the Easysoft ODBC (Welcome) dialog box appears.

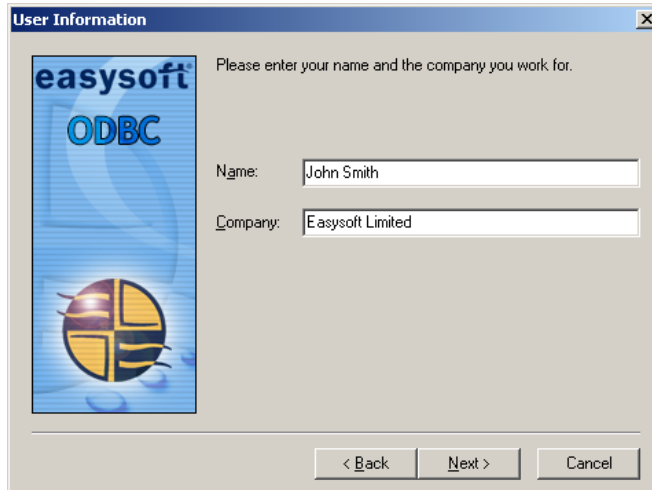


Click Next to continue. The Software License Agreement dialog box is displayed.



After you have read the License Agreement, click Yes to confirm your acceptance of its terms. If you do not accept the terms of the agreement, click No and then click Exit Setup to exit the installation.

The User Information dialog box is displayed.



Type your name and company in the spaces provided. Click Next.

In the Start Copying Files dialog box, click Next.



In the Setup Complete dialog box, click Finish to return to Windows. To display the Easysoft ODBC Driver online Help, click the check box.



The installation successfully terminates, and the default Microsoft ODBC Administrator icon is created. Typically, the ODBC Administrator icon can be found in the Control Panel, or under the Start menu.

APPENDIX E

Easysoft COBOL Converter

The Easysoft COBOL Converter is used to convert the file definitions contained in the FILE SECTION of COBOL files to the CSV form (Comma Separated Value) required by the PC-based Easysoft Administrator and the server-based Administrator.

The Easysoft Cobol Converter conforms to the following COBOL standards:

- ANSI '74
- ANSI '85 (ANSI X3.23 - 1985)
- X/Open
- VAX COBOL 5.3
- DEC COBOL 2.4

This document is based on the assumption that you know about the form and function of the FILE SECTION of COBOL files.

The COBOL converter is not contained within the basic the Easysoft ODBC for RMS software. For the installation and use of the COBOL converter, you will need write privileges to the server.

Installation

The software is supplied as a compressed file (one per platform) on the Easysoft Web Site (<http://www.easysoft.com/>).

The installed files on the RMS platform are approximately 170 Kbytes (the size varies slightly, depending on whether the hardware platform is VAX or Alpha).

Any user with write privileges on the server can install the converter.

1. Download the zipped file from the Easysoft Web Site onto your PC. Place the file in any convenient temporary location.
2. Transfer the zipped file from the PC to your server. This must be done using binary transfer mode.

The zipped file should be placed in the directory which is referenced by the EASYSOFT_SQL_SYSTEM logical. This logical will have been set up during the installation of Easysoft ODBC for RMS, and by default is [EASYSOFT.SQL.SYSTEM]. It is possible that the Easysoft software was installed in a different physical location. To ascertain where the EASYSOFT_SQL_SYSTEM logical points to, at the command prompt type:

```
$ SHOW LOG EASYSOFT_SQL_SYSTEM
```

The output will contain the line:

```
"EASYSOFT_SQL_SYSTEM" = "<device>:[<directory>]"
```

- Unzip the file (a copy of the required executable resides in EASYSOFT_SQL_SYSTEM).

To do this type:

```
$ UNZIP <filename>
```

The following files will be generated:

COBOLCNV.OLB, COBOLCNV.COM, COBOLSETUP.COM

- Run the setup routine:

```
$ @COBOLSETUP
```

The following file will be generated: COBOLCNV.EXE

The COBOLCNV.OLB file will be removed.

The software can now be used.

- Remove the zipped file from the PC and COBOLSETUP.COM from the server (this step is optional).

Using the Converter

This section describes the options available; examples for each option are presented after this.

The entire process of conversion and using the generated converted file is:

- Convert the COBOL file (all file and record definitions in the FD section are converted)
- Either
 - Import the CSV file directly into the Easysoft Administrator on the PC (this import will include a validation process), or
 - Import the file in two stages. First, the CSV file should be imported into the server-based administrator, then download the catalog into the PC Administrator. The advantage of the two-stage process is that with large files, the overall import time will be quicker than by importing the CSV file directly into the PC-based Administrator
- Set the database privileges (if the imported database is new)
- Upload the definitions to the catalog on the server

The command syntax to run the converter is:

```
$ @EASYSOFT_SQL_SYSTEM:COBOLCNV <COBOL file specification> [<options>]
```

Where <COBOL file specification> is the name of the COBOL file to convert. This name must include any file extension. If this parameter is omitted, a short listing of the options available is displayed, and then the routine stops.

<options> is one or more of the following, in any order.

Option	Explanation
/OUT=output_file	<output_file> is the name of the file which is generated by the conversion process. If <output_file> is not specified, then the output is sent to the screen.
/FILLERS	Outputs FILLER fields.
/SUBSTRUCTURES	Outputs group items (sub-structured fields).
/TURNON	Converts raw record descriptions (e.g. from a text file containing these).

After the COBOL file has been parsed, you are asked to supply the name of the database and the default directory. See “Basic Conversion”, page 32, for details.

There is no limit to the size of the COBOL file which can be converted.

The converter has no effect on any other software.

If you wish to quit the routine whilst a conversion is taking place, then do the following:

Press **ctrl-y**

If the FILE SECTION of the COBOL file contains invalid syntax, then a message is generated stating this, but processing will continue whenever possible, so that those definitions which are valid will be converted.

Basic Conversion

This example shows the basic options required to convert a COBOL file. Say there is a COBOL file called TESTDATA.TXT. To convert this and put the output in a file called TESTDATA.CSV, you would type:

```
$ @EASYSOFT_SQL_SYSTEM:COBOLCNV TESTDATA.TXT /OUT=TESTDATA.CSV
```

```
Easysoft COBOL converter Version <version>  
<copyright notice>
```

```
Processing file  
Parsing COBOL data
```

Database Name : **TEST**
 Default Directory : **TESTDIR**

After an introductory message, you are asked for Database Name. This is the name that you want to call the database which holds the definitions which will be generated by the conversion. You must enter a value.

Default directory refers to the default location of the files stored on the server. It is optional; if a value is not entered, an empty string is generated in the output.

```

Converting parsed data to CSV data.
Conversion Statistics
Total number of FILE definitions           : <n>
<other conversion statistics
Conversion completed successfully
  
```

Comparison

Say the COBOL file contained these definitions:

```

FD EMPLOYEES
  <other definitions may appear here>

01 EMPLOYEE.
   03 SSN                PIC X(9).
   03 NAME                PIC X(30).
  
```

The CSV file that is generated would contain the following:

```

"FILE", "EMPLOYEES", "INDEXED", "FIXED", 39
"FIELD", "EMPLOYEES", "SSN", "STRING", 0, 9, 9, 0, 0, "", ""
"FIELD", "EMPLOYEES", "NAME", "STRING", 9, 30, 30, 0, 0, "", ""
"DB", "TEST", "TESTDIR", "<driver>", ""
"TABLE", "TEST", "EMPLOYEE", "TABLE", "EMPLOYEES", "EMPLOYEES", ""
"COLUMN", "TEST", "EMPLOYEE", "SSN", "SSN", "VARCHAR", 9, 1, 1, ""
"COLUMN", "TEST", "EMPLOYEE", "NAME", "NAME", "VARCHAR", 30, 1, 1, ""
  
```

*This is the database name
that was entered at the start
of the processing.*

*This is the default directory that was
entered at the start of the processing.*

The Database definition (4th line of the generated CSV) contains the name of the database and default directory (if any) that you entered. The name of the driver depends upon the platform; if you are using Easysoft ODBC for RMS, the driver name will be VAX-RMS.

Any hyphen characters in file or record definitions in the COBOL file (File or Field definitions in the Easysoft Administrator) are converted to underscore characters in the Table and Column definitions in the Easysoft Administrator.

Output FILLER Fields

By default, FILLER fields and fields beginning with FILLER- are not converted. If you want to see the FILLER fields, then use this option.

Because field names in the Easysoft Administrator (and corresponding SQL column names) must be unique within a record (row), a suffix is added to the FILLER fields. This suffix is `_n`, where `n` represents a character digit corresponding to the additional FILLER field.

Here is a very contrived example. Say there was a record, FILLDEMO, defined as follows:

```
01 FILLDEMO.
 03 FIRSTNAME          PIC X(20).
 03 FILLER              PIC XXXX.
 03 LASTNAME           PIC X(20).
 03 FILLER              PIC X(12).
 03 FILLER              PIC X(16).
 03 FILLER-NUMERIC     PIC 99.
```

Without this option, the resulting column definitions would be:

```
"COLUMN", "TEST", "FILLDEMO", "FIRSTNAME", "FIRSTNAME", "VARCHAR", 20, 1, 1, " "
"COLUMN", "TEST", "FILLDEMO", "LASTNAME", "LASTNAME", "VARCHAR", 20, 1, 1, " "
```

To see all the FILLER fields, use the option, thus:

```
$ @EASYSOFT_SQL_SYSTEM:COBOLCNV TESTDATA.TXT /OUT=TESTDATA.CSV /FILLERS
```

Using this option, the resulting column definitions for the FILLDEMO record would be:

```
"COLUMN", "TEST", "FILLDEMO", "FIRSTNAME", "FIRSTNAME", "VARCHAR", 20, 1, 1, " "
"COLUMN", "TEST", "FILLDEMO", "FILLER_1", "FILLER_1", "VARCHAR", 4, 1, 1, " "
"COLUMN", "TEST", "FILLDEMO", "LASTNAME", "LASTNAME", "VARCHAR", 20, 1, 1, " "
"COLUMN", "TEST", "FILLDEMO", "FILLER_2", "FILLER_2", "VARCHAR", 12, 1, 1, " "
"COLUMN", "TEST", "FILLDEMO", "FILLER_3", "FILLER_3", "VARCHAR", 16, 1, 1, " "
"COLUMN", "TEST", "FILLDEMO", "FILLER_NUMERIC", "FILLER-NUMERIC", "VARCHAR", 2, 1, 1, " "
```

Fields defined as numeric in COBOL (e.g. PIC 99) are converted to character fields, because the NUMERIC DataTips consists of digit characters.

Output Group Items (sub-structured fields)

If the COBOL file contains a hierarchical record structure (sub-structured fields) only the lowest level fields (elementary items) are converted by default.

For example, you may have the following COBOL substructure:


```

FD EMPLOYEES
  <other definitions may appear here>

01 EMPLOYEE.
  03 SSN          PIC X(9).
  03 NAME.
    05 FIRSTNAME  PIC X(14).
    05 LASTNAME   PIC X(16).

```

Visually, this equates to:

SSN	NAME	
	FIRST NAME	LAST NAME

Without this option only the lowest level is converted.

By default, the output from the conversion process would be:

```

"FILE", "EMPLOYEES", "INDEXED", "FIXED", 39
"FIELD", "EMPLOYEES", "SSN", "STRING", 0, 9, 9, 0, 0, "", ""
"FIELD", "EMPLOYEES", "FIRSTNAME", "STRING", 9, 14, 14, 0, 0, "", ""
"FIELD", "EMPLOYEES", "LASTNAME", "STRING", 23, 16, 16, 0, 0, "", ""
"DB", "TEST", "TESTDIR", "VAX-RMS", ""
"TABLE", "TEST", "EMPLOYEE", "TABLE", "EMPLOYEES", "EMPLOYEES", ""
"COLUMN", "TEST", "EMPLOYEE", "SSN", "SSN", "VARCHAR", 9, 1, 1, ""
"COLUMN", "TEST", "EMPLOYEE", "FIRSTNAME", "FIRSTNAME", "VARCHAR", 14, 1, 1, ""
"COLUMN", "TEST", "EMPLOYEE", "LASTNAME", "LASTNAME", "VARCHAR", 16, 1, 1, ""

```

The NAME field and its corresponding columns does not appear. To see the NAME field, you would use the option, thus:

```
$ @EASYSOFT_SQL_SYSTEM:COBOLCNV TESTDATA.TXT /OUT=TESTDATA.CSV /SUBSTRUCTURES
```

The output (only table and column definitions shown) would then be:

```

"TABLE", "TEST", "EMPLOYEE", "TABLE", "EMPLOYEES", "EMPLOYEES", ""
"COLUMN", "TEST", "EMPLOYEE", "SSN", "SSN", "VARCHAR", 9, 1, 1, ""
"COLUMN", "TEST", "EMPLOYEE", "NAME", "NAME", "VARCHAR", 30, 1, 1, ""
"COLUMN", "TEST", "EMPLOYEE", "FIRSTNAME", "FIRSTNAME", "VARCHAR", 14, 1, 1, ""
"COLUMN", "TEST", "EMPLOYEE", "LASTNAME", "LASTNAME", "VARCHAR", 16, 1, 1, ""

```

NAME appears in output.

Convert Raw Record Descriptions

If you have a file containing valid COBOL record descriptions, and if that file is not part of a COBOL program proper, you can still convert the record descriptions. Say you had a text file which contained just this data:

```

01 EMPLOYEE.
  03 SSN          PIC X(9).
  03 NAME         PIC X(30).

```

you would use this option to convert it, thus:

```
$ @EASYSOFT_SQL_SYSTEM:COBOLCNV TESTDATA.TXT /OUT=TESTDATA.CSV /TURNON
```

The output would be:

```
"FILE", "EMPLOYEES", "INDEXED", "FIXED", 39  
"FIELD", "EMPLOYEES", "SSN", "STRING", 0, 9, 9, 0, 0, "", ""  
"FIELD", "EMPLOYEES", "NAME", "STRING", 9, 30, 30, 0, 0, "", ""  
"DB", "TEST", "TESTDIR", "<driver>", ""  
"TABLE", "TEST", "EMPLOYEE", "TABLE", "EMPLOYEES", "EMPLOYEES", ""  
"COLUMN", "TEST", "EMPLOYEE", "SSN", "SSN", "VARCHAR", 9, 1, 1, ""  
"COLUMN", "TEST", "EMPLOYEE", "NAME", "NAME", "VARCHAR", 30, 1, 1, ""
```

Notes / Information

If the COBOL file contains multiple definitions implemented using the REDEFINES or RENAMES keywords, then after import into the Easysoft Administrator the user should deselect columns from the relevant tables to prevent multiple views of the same data in an ODBC application.

Database privileges are not set, so the user is required to set them using the Easysoft Administrator before any data can be accessed.

There is no un-install option for the software. To remove the software, delete the compressed file that was transferred to the server and the files that were generated during the installation.

COBOL Data Types

COBOL PIC	Size of n	COBOL USAGE	Number of bytes	VMS Data type	Easysoft Field	Easysoft Column
PIC X(n)	n <= 65,535	USAGE IS DISPLAY	n	ASCII Text	STRING	CHAR
PIC S9(n)	n <= 18	USAGE IS DISPLAY SIGN IS TRAILING	n	Trailing Overpunched Numeric	TRAILING- OVERPUNCHED- NUMERIC	CHAR
PIC S9(n)	n <= 18	USAGE IS DISPLAY SIGN IS LEADING	n	Leading Overpunched Numeric	LEADING- OVERPUNCHED- NUMERIC	CHAR
PIC S9(n)	n <= 18	USAGE IS DISPLAY SIGN IS TRAILING SEPARATE	n+1	Trailing Separate Numeric	TRAILING-SIGN- NUMERIC	CHAR
PIC S9(n)	n <= 18	USAGE IS DISPLAY SIGN IS LEADING SEPARATE	n+1	Leading Separate Numeric	LEADING-SIGN- NUMERIC	CHAR
PIC 9(n)	n <= 18	USAGE IS DISPLAY	n	Unsigned Numeric	NUMERIC	CHAR
PIC 9(n)	n <= 4	USAGE IS COMP	2	Word Integer	WORD	SMALLINT
PIC 9(n)	5 <= n <= 9	USAGE IS COMP	4	Longword Integer	LONGWORD	INTEGER
PIC 9(n)	10 <= n <= 18	USAGE IS COMP	8	Quadword Integer	QUADWORD	DOUBLE
PIC S9(n)	n <= 4	USAGE IS COMP	2	Word Integer	WORD	SMALLINT
PIC S9(n)	5 <= n <= 9	USAGE IS COMP	4	Longword Integer	LONGWORD	INTEGER

COBOL PIC	Size of n	COBOL USAGE	No. Bytes	VMS Data type	Easysoft Field	Easysoft Column
PIC S9(n)	10 <= n <= 18	USAGE IS COMP	8	Quadword Integer	QUADWORD	DOUBLE
		USAGE IS INDEX	4	Longword Integer	LONGWORD	INTEGER
		USAGE IS POINTER	4	Longword Integer	LONGWORD	INTEGER
		USAGE IS COMP-1	4	F_floating	F-FLOATING	DOUBLE
		USAGE IS COMP-2	8	D_floating	D-FLOATING	DOUBLE
PIC S9(n)	n <= 8	USAGE IS COMP-3	(n+1)/2 rounded up	Packed Decimal	PACKED-S	CHAR
PIC 9(n)	n <= 18	USAGE IS COMP-3	(n+1)/2 rounded up	Packed Decimal	PACKED-U	CHAR
PIC S9(n)V9(s)	(n+s) <= 18	USAGE IS DISPLAY	n+s	Trailing Overpunched Numeric	TRAILING-OVERPUNCHED-NUMERIC SCALE=s	CHAR
PIC S9(n)V9(s)	(n+s) <= 18	USAGE IS DISPLAY SIGN IS TRAILING	n+s	Trailing Overpunched Numeric	TRAILING-OVERPUNCHED-NUMERIC SCALE=s	CHAR
PIC S9(n)V9(s)	(n+s) <= 18	USAGE IS DISPLAY SIGN IS LEADING	n+s	Leading Overpunched Numeric	LEADING-OVERPUNCHED-NUMERIC SCALE=s	CHAR
PIC S9(n)V9(s)	(n+s) <= 18	USAGE IS DISPLAY SIGN IS TRAILING SEPARATE	n+s+1	Trailing Separate Numeric	TRAILING-SIGN-NUMERIC	CHAR
PIC S9(n)V9(s)	(n+s) <= 18	USAGE IS DISPLAY SIGN IS LEADING SEPARATE	n+s+1	Leading Separate Numeric	LEADING-SIGN-NUMERIC	CHAR

COBOL PIC	Size of n	COBOL USAGE	No. Bytes	VMS Data type	Easysoft Field	Easysoft Column
PIC 9(n)V9(s)	$(n+s) \leq 18$	USAGE IS DISPLAY	n+s	Unsigned Numeric	NUMERIC SCALE=s	CHAR
PIC 9(n)V9(s)	$(n+s) \leq 4$	USAGE IS COMP	2	Word Integer	WORD SCALE=s	SMALLINT
PIC 9(n)V9(s)	$5 \leq (n+s) \leq 9$	USAGE IS COMP	4	Longword Integer	LONGWORD SCALE=s	INTEGER
PIC 9(n)V9(s)	$10 \leq (n+s) \leq 18$	USAGE IS COMP	8	Quadword Integer	QUADWORD SCALE=s	DOUBLE
PIC S9(n)V9(s)	$(n+s) \leq 4$	USAGE IS COMP	2	Word Integer	WORD SCALE=s	SMALLINT
PIC S9(n)V9(s)	$5 \leq (n+s) \leq 9$	USAGE IS COMP	4	Longword Integer	LONGWORD SCALE=s	INTEGER
PIC S9(n)V9(s)	$10 \leq (n+s) \leq 18$	USAGE IS COMP	8	Quadword Integer	QUADWORD SCALE=s	DOUBLE
PIC 9(n)V9(s)	$(n+s) \leq 18$	USAGE IS COMP-3	$(n+s+1)/2$ rounded up	Packed Decimal	PACKED-U SCALE=s	CHAR
PIC S9(n)V9(s)	$(n+s) \leq 18$	USAGE IS COMP-3	$(n+s+1)/2$ rounded up	Packed Decimal	PACKED-S SCALE=s	CHAR

APPENDIX F

Easysoft PowerHouse PDL Converter

The Easysoft PowerHouse PDL Converter is used to convert PowerHouse data dictionary file definitions to the CSV form (Comma Separated Value) required by the PC-based Easysoft Administrator and the server-based Administrator. These file definitions are stored in PowerHouse Dictionary Language files, which are also known as PDL files.

Since QSHOW, which is part of the PowerHouse package, generates PDL files, the converter is also known as the QSHOW converter.

This document is based on the assumption that you know about the form and function of PDL files.

For the installation and use of the PDL converter, you will need write privileges to the server.

Using the Converter

This section describes the options available; examples for each option are presented after this.

The entire process of conversion and using the generated CSV file is:

1. Convert the PDL file
2. Either
 - a) Import the CSV file directly into the Easysoft Administrator on the PC (this import will include a validation process), or
 - b) Import the file in two stages. First, the CSV file should be imported into the server-based administrator, then download the catalog into the PC Administrator. The advantage of the two-stage process is that with large files, the overall import time will be quicker than by importing the CSV file directly into the PC-based Administrator
3. Set the database privileges (if the imported database is new)
4. Upload the definitions to the catalog on the server

The command syntax to run the converter is:

```
$ @EASYSOFT_SQL_SYSTEM:PDLCONVERT <PDL file specification> [<options>]
```

Where <PDL file specification> is the name of the PDL file to convert. This name must include the PDL file extension (if it exists). If this parameter is omitted, a short listing of the options available is displayed, and then the routine stops.

<options> is one or more of the following, in any order.

Option	Explanation
/OUT=output_file	<p><output_file> is the name of the file which is generated by the conversion process.</p> <p>If <output_file> is not specified, then the output is sent to the screen.</p>
/FILLERS	Outputs multiple FILLER and FILLER-NUMERIC fields.
/SUBSTRUCTURES	Outputs sub-structured fields.
/DEL=delete_chars	Prevents the specified characters from being generated in the output. Used in cases where the PDL definitions contain characters which are not allowed in the Easysoft CSV definitions.
/DEF=define	<p>Define condition names for use by the pre-processor.</p> <p>If condition variables are defined in the PDL file, you may want to convert the sections of the file to correspond with these condition variables.</p>

After the PDL file has been parsed, you are asked to supply the name of the database and the default directory. See “Basic Conversion”, page 32, for details.

There is no limit to the size of the PDL file which can be converted.

The converter has no effect on any other software.

If you wish to quit the routine whilst a conversion is taking place, then do the following:

Press **Ctrl-Y**.

If the PDL file contains invalid PDL syntax, then a message is generated stating this, but processing will continue whenever possible, so that those definitions which are valid will be converted.

Basic Conversion

This example shows the basic options required to convert a PDL file. Say there is a PDL file called TESTDATA.PDL. To convert this and put the output in a file called TESTDATA.CSV, you would type:

```
$ @EASYSOFT_SQL_SYSTEM:PDLCONVERT TESTDATA.PDL /OUT=TESTDATA.CSV
```

```
Easysoft PowerHouse PDL Converter Version <version>
<copyright notice>
```

```
Pre-processing file
Processing file
Parsing PDL data
```

```
Database Name : TEST
```

```
Default Directory : TESTDIR
```

After an introductory message, you are asked for Database Name. This is the name that you want to call the database which holds the definitions which will be generated by the conversion. You must enter a value.

Default directory refers to the location of the data files stored on the server . It is optional; if a value is not entered, an empty string is generated in the output.

```
Converting parsed data to CSV data.
```

```
Conversion Statistics
```

```
Total number of FILE definitions          : <n>
<other conversion statistics>
```

```
Conversion completed successfully
```

Comparison

Say the PDL file contained these definitions:

```
File EMPLOYEES  Organization INDEXED  CREATE
```

```
Record EMPLOYEE
```

```
Item SSN      Datatype CHARACTER  Size 9
Item NAME     Datatype CHARACTER  Size 30
```

The CSV file that is generated would contain the following:


```
"FILE", "EMPLOYEES", "INDEXED", "FIXED", 39
"FIELD", "EMPLOYEES", "SSN", "STRING", 0, 9, 9, 0, 0, "", ""
"FIELD", "EMPLOYEES", "NAME", "STRING", 9, 30, 30, 0, 0, "", ""
"DB", "TEST", "TESTDIR", "VAX-RMS", ""
"TABLE", "TEST", "EMPLOYEE", "TABLE", "EMPLOYEES", "EMPLOYEES", ""
"COLUMN", "TEST", "EMPLOYEE", "SSN", "SSN", "VARCHAR", 9, 1, 1, ""
"COLUMN", "TEST", "EMPLOYEE", "NAME", "NAME", "VARCHAR", 30, 1, 1, ""
```

This is the database name that was entered at the start of the processing.

This is the default directory that was entered at the start of the processing.

The Database definition (4th line) contains the name of the database and default directory (if any) that you entered.

Any hyphen characters in Record or Item definitions in the PDL file (File or Field definitions in the Easysoft Administrator) are converted to underscore characters in the Table and Column definitions in the Easysoft Administrator.

Output FILLER Fields

If a record contains more than one FILLER field, then by default, only the first one is converted. If you want to see all the FILLER fields, then use this option.

FILLER_NUMERIC fields, and any other fields starting with the name FILLER- (hyphen) or FILLER_ (underscore) are not converted unless this option is used.

Because field names in the Easysoft Administrator (and corresponding SQL column names) must be unique within a record (row), a suffix is added to the second and subsequent FILLER fields. This suffix is _n, where n represents a character digit corresponding to the additional FILLER field.

Here is a contrived example. Say there was a record, FILLDEMO, defined as follows:

```
Record FILLDEMO
Item FIRSTNAME      Datatype CHARACTER   Size 20
Item FILLER         Datatype CHARACTER   Size 4
Item LASTNAME      Datatype CHARACTER   Size 20
Item FILLER        Datatype CHARACTER   Size 12
Item FILLER        Datatype CHARACTER   Size 16
Item FILLER-NUMERIC Datatype INTEGER     Size 2
Item FILLER-NUMERIC Datatype INTEGER     Size 2
```

Without this option, the resulting column definitions would be:

```
"COLUMN", "TEST", "FILLDEMO", "FIRSTNAME", "FIRSTNAME", "VARCHAR", 20, 1, 1, ""
"COLUMN", "TEST", "FILLDEMO", "FILLER", "FILLER", "VARCHAR", 4, 1, 1, ""
"COLUMN", "TEST", "FILLDEMO", "LASTNAME", "LASTNAME", "VARCHAR", 20, 1, 1, ""
```

To see all the FILLER fields, use the option, thus:

```
$ @EASYSOFT_SQL_SYSTEM:PDLCONVERT TESTDATA.PDL /OUT=TESTDATA.CSV
/FILLERS
```

Using the option, the resulting column definitions for the FILLDEMO record would be:

```
"COLUMN", "TEST", "FILLDEMO", "FIRSTNAME", "FIRSTNAME", "VARCHAR", 20, 1, 1, ""
"COLUMN", "TEST", "FILLDEMO", "FILLER", "FILLER", "VARCHAR", 4, 1, 1, ""
"COLUMN", "TEST", "FILLDEMO", "LASTNAME", "LASTNAME", "VARCHAR", 20, 1, 1, ""
"COLUMN", "TEST", "FILLDEMO", "FILLER_1", "FILLER_1", "VARCHAR", 12, 1, 1, ""
"COLUMN", "TEST", "FILLDEMO", "FILLER_2", "FILLER_2", "VARCHAR", 16, 1, 1, ""
"COLUMN", "TEST", "FILLDEMO", "FILLER_NUMERIC", "FILLER-
NUMERIC", "SMALLINT", 2, 1, 1, ""
"COLUMN", "TEST", "FILLDEMO", "FILLER_NUMERIC_1", "FILLER-
NUMERIC_1", "SMALLINT", 2, 1, 1, ""
```

Output Sub-structured Fields

If the PDL file contains sub-structured items implemented using the BEGIN STRUCTURE and END STRUCTURE elements these are not converted by default. Only the lowest level fields are converted.

For example, you may have the following PDL substructure:

```
File EMPLOYEES      Organization INDEXED CREATE

Record EMPLOYEE
  Item SSN          Datatype CHARACTER   Size 9
  Item NAME        Datatype CHARACTER   Size 30
  BEGIN STRUCTURE
    Item FIRSTNAME Datatype CHARACTER   Size 14
    Item LASTNAME  Datatype CHARACTER   Size 16
  END STRUCTURE
```

Visually, this equates to:

SSN	NAME	
	FIRST NAME	LAST NAME

Without this option only the lowest level is converted.

By default, the output from the conversion process would be:

```
"FILE", "EMPLOYEES", "INDEXED", "FIXED", 39
"FIELD", "EMPLOYEES", "SSN", "STRING", 0, 9, 9, 0, 0, "", ""
"FIELD", "EMPLOYEES", "FIRSTNAME", "STRING", 9, 14, 14, 0, 0, "", ""
"FIELD", "EMPLOYEES", "LASTNAME", "STRING", 23, 16, 16, 0, 0, "", ""
"DB", "TEST", "TESTDIR", "VAX-RMS", ""
"TABLE", "TEST", "EMPLOYEE", "TABLE", "EMPLOYEES", "EMPLOYEES", ""
"COLUMN", "TEST", "EMPLOYEE", "SSN", "SSN", "VARCHAR", 9, 1, 1, ""
"COLUMN", "TEST", "EMPLOYEE", "FIRSTNAME", "FIRSTNAME", "VARCHAR", 14, 1, 1, ""
"COLUMN", "TEST", "EMPLOYEE", "LASTNAME", "LASTNAME", "VARCHAR", 16, 1, 1, ""
```

The NAME field and its corresponding columns does not appear. To see the NAME field, you would use the option, thus:

```
$ @EASYSOFT_SQL_SYSTEM:PDLCONVERT TESTDATA.PDL /OUT=TESTDATA.CSV
/SUBSTRUCTURES
```

The output (only table and column definitions shown) would then be:

```
"TABLE", "TEST", "EMPLOYEE", "TABLE", "EMPLOYEES", "EMPLOYEES", " "
"COLUMN", "TEST", "EMPLOYEE", "SSN", "SSN", "VARCHAR", 9, 1, 1, " "
"COLUMN", "TEST", "EMPLOYEE", "NAME", "NAME", "VARCHAR", 30, 1, 1, " "
"COLUMN", "TEST", "EMPLOYEE", "FIRSTNAME", "FIRSTNAME", "VARCHAR", 14, 1, 1, " "
"COLUMN", "TEST", "EMPLOYEE", "LASTNAME", "LASTNAME", "VARCHAR", 16, 1, 1, " "
```

*NAME appears
in output.*

Stripping Characters

Characters which are valid in the PDL definition may not be valid in the Easysoft CSV format, for example, the \$ character.

Valid characters in the Easysoft Administrator are:

File Name	a to z, A to Z, underscore (_), hyphen (-)
Field Name	a to z, A to Z, underscore (_), hyphen (-)
Database Name	a to z, A to Z
Table Name	a to z, A to Z, underscore (_)
Column Name	a to z, A to Z, underscore (_)

To prevent invalid characters from being generated in the output file, use the option thus:

```
$ @EASYSOFT_SQL_SYSTEM:PDLCONVERT TESTDATA.PDL /OUT=TESTDATA.CSV /DEL=&
```

In the case exemplified above, the ampersand character (&), if it existed in the PDL record or item definitions, would not appear in the CSV output.

PDL definitions contain a section called System Options, and within this, there may be a section entitled Special Name Characters. Any characters that are listed in Special Name Characters will not appear in the output file. There is no need to use this option to remove them.

```
System Options                                     &
<definitions>                                     &
Special Name Characters "-_ '%#" _____ & _____ These characters will not
<definitions>                                     & appear in the output file.
```

Define Conditions for use by the Pre-processor

Say that the following record structure is defined in PDL.

```
File NAMES      Organization INDEXED CREATE
@IF GB
  Record NAME
    Item FIRSTNAME      Datatype CHARACTER      Size 18
    Item LASTNAME       Datatype CHARACTER      Size 20
@ELSEIF FR
  Record NOM
    Item PRENOM         Datatype CHARACTER      Size 18
    Item NOM            Datatype CHARACTER      Size 20
@ENDIF
```

If the converter is used without this option, then the output will not contain information for either the record called NAME, or the record called NOM (because neither of the condition variables is set to TRUE). If you ran the converter, you would see the following definitions only:

```
"FILE", "NAMES", "INDEXED", "FIXED", 1
"DB", "TEST", "TESTDIR", "VAX-RMS", ""
```

To convert, for example, the French version (condition variable FR), you would use this option to set the condition variable, thus:

```
$ @EASYSOFT_SQL_SYSTEM:PDLCONVERT TESTDATA.PDL /OUT=TESTDATA.CSV /DEF=FR
```

The output would be:

```
"FILE", "NAMES", "INDEXED", "FIXED", 38
"FIELD", "NAMES", "PRENOM", "STRING", 0, 18, 18, 0, 0, "", ""
"FIELD", "NAMES", "NOM", "STRING", 18, 20, 20, 0, 0, "", ""
"DB", "TEST", "TESTDIR", "VAX-RMS", ""
"TABLE", "TEST", "NOM", "TABLE", "NAMES", "NAMES", ""
"COLUMN", "TEST", "NOM", "PRENOM", "PRENOM", "VARCHAR", 18, 1, 1, ""
"COLUMN", "TEST", "NOM", "NOM", "NOM", "VARCHAR", 20, 1, 1, ""
```

You can set as many condition variables as you like; each one must be preceded by the -d option. The order in the command line is not important. The output depends upon the structure of the PDL file; output is based on the first condition that is TRUE. Say there were three conditions:

```
@IF GB OR US
  <definition 1>
@ELSEIF FR
  <definition 2>
@ELSEIF DM
  <definition 3>
@ENDIF
```

If the define options contained: -d FR -d US, then <definition 1> would be converted.

Notes / Information

If the PDL file contains multiple defined fields implemented using the REDEFINES keyword in an ITEM statement, then after importation into the Easysoft Administrator the user should deselect columns from the relevant tables to prevent multiple views of the same data appearing in an ODBC application.

Database privileges are not set, so the user is required to set them using the Easysoft Administrator before any data can be accessed.

There is no un-install option for the software. To remove the software, delete the compressed file that was transferred to the server and the files that were generated during the installation.

APPENDIX G

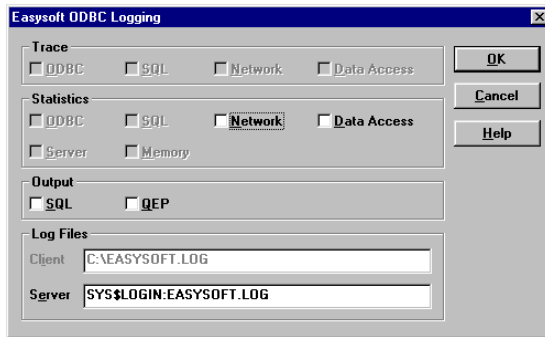
Troubleshooting

This appendix shows you how to investigate problems that may occur, and what to do to resolve them. You will see how to:

- determine what ODBC function calls are being sent to the server
- check what is happening to the network
- see what physical files are accessed by a query and the number of I/Os for that query
- view the SQL that is sent to the server
- view the Query Execution Plan for a query
- use the Microsoft ODBC Administrator to trace ODBC function calls
- work out what is happening at the server (processes running and how to stop them)

Easysoft ODBC Logging

The Easysoft ODBC Logging dialog box is used to specify what is logged and where the log information is saved. It is accessed by selecting **Logging...** on the Easysoft ODBC Setup dialog box.



Easysoft ODBC Logging Information

(Only available options are described)

Trace ODBC	The ODBC functions that are used are recorded in the log file. The option can only be enabled using the EASYSOFT.INI file, - see the next section.
Trace Network	Network information is recorded in the log file. The option can only be enabled using the EASYSOFT.INI file (described below this table).
Network (Statistics)	Statistics about data transmitted and received.
Data Access (Statistics)	Lists the files accessed for a given query and the number of I/Os to files for that query.
SQL (Output)	Lists the SQL query which is passed to the Server.
QEP (Output)	Lists the Query Execution Plan.
Log file (Server)	Path and name of server file to which logging information is sent. Default can be changed.

EASYSOFT.INI

The Easysoft Administrator installation process places new files and modifies existing files in the Windows directory. Of relevance is the EASYSOFT.INI file, which is described here.

This file controls the Easysoft Administrator and Easysoft ODBC. An example is shown below, and following this, some important settings/options are described in detail.

```
[Options]
SystemDB=C:\EASYSOFT\SQL\SYSTEM\essystem.sec
[Administrator]
Path=C:\EASYSOFT\SQL\SYSTEM
Language=ENG
Username=Mike
Company=Easysoft Limited

[LOGGING]
ODBC=1
NET=1
```

Explanation of Sections

The order of the sections in the EASYSOFT.INI file is not important.

Comments can be included by prefixing a line with a semicolon character (;).

[LOGGING]

ODBC=1 To turn on the **Trace ODBC** option in the Easysoft ODBC Logging dialog box, set this value to 1. The log file that is generated is called \ESODBC.LOG.

NET=1 To turn on the **Trace Network** option in the Easysoft ODBC Logging dialog box, set this value to 1. The log file that is generated is called \ESNET.LOG.

[NETWORK]

IgnoreWinsock=1 If Pathworks is being used, setting this flag to 1 disables Winsock.

[SETTINGS]

messages=0 To turn off informational messages generated by the Server.

[Options]

AllowReservedWords=1 This option lets you use words reserved for SQL(see “Reserved Words” in Module 2. The default is 0 (i.e. the line can be omitted).

[Administrator]

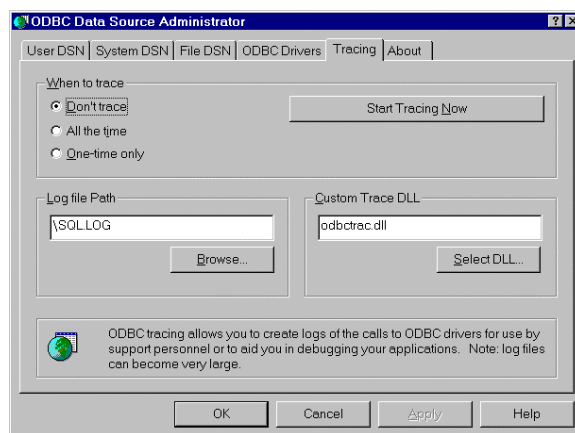
This section contains information relating to the Administrator setup.

Path	Location of Easysoft files.
Language	Defines the language; currently only English is supported.
Username	The name of the user as specified in the Easysoft Administrator Setup (User Details) dialog box.
Company	The name of the company as specified in the Easysoft Administrator Setup (User Details) dialog box.
DatabaseRepair=True	Repairs corrupt local database.
ThreeD=False	The Easysoft Administrator uses the Microsoft file CTL3D.DLL which gives a 3-D effect to Windows dialog boxes. Sometimes, other applications can interfere with the correct functioning of this. One solution is to not use the file by: ThreeD=False.

Microsoft Trace Options

Version 3.0 Trace Options

The Tracing tab is used to specify how the ODBC Driver Manager traces calls to ODBC functions. The information that is generated using the Tracing tab is not particularly useful to the average user, but if there are problems with your system we may request that you send us a log file.



When to trace

These options can only be set when there is no connection.

Don't trace: disables tracing.

All the time: tracing is performed for all connections at all times.

One-time only: tracing is performed for the next connection, then disabled.

Start Tracing Now

Enables dynamic tracing, which is performed as long as the dialog box is open. Dynamic tracing can be enabled whether or not a connection is open. When this option is selected, the button is replaced with Stop Tracing Now. When Stop Tracing Now is selected, or when the ODBC Administrator dialog box is closed, dynamic tracing is disabled.

Log file Path

Displays the path and file where the log information will be stored. You can change the path and file name by editing the entry box, or by using the **Browse** button.

Custom Trace DLL

If you prefer to use your own custom DLL to perform the tracing operation, replace the default file with the name of your file.

Click the **Apply** button to accept changes without closing the dialog box, or click **OK** to make changes and close the dialog box.

SQL.LOG Described

```
SQLAllocEnv(phenv0006E740);
SQLAllocConnect(henv0006E740, phdbc0007B658);
SQLDriverConnect(hdbc0007B658, hwnd00010456, "DSN=Coda - Fulla - User
Carolyn;UID=MIKEU", 41, szConnStrOut, 256, pcbConnStrOut, 1);
SQLGetInfo(hdbc0007B658, 2, rgbInfoValue, 256, pcbInfoValue);
SQLAllocStmt(hdbc0007B658, phstmt000A3AB0);
SQLGetInfo(hdbc0007B658, 1, rgbInfoValue, 2, pcbInfoValue);
```

The log file that is generated lists the ODBC functions that were called by the query. This file can become huge. For example, during a test performed when this course was developed, the SQL.LOG grew to over 14 Mbytes.

How to Work Out What the Server is Doing

Imagine this scenario. You've started a query from some application on the PC, and the hourglass symbol appears. You wait. And wait... You decide you don't want to wait any longer. But what can you do?

In this section, you will see how to determine what processes are running on the server and how to stop the process that is causing the long wait at the PC. The overall steps are:

1. log on to server
2. determine what process is causing the problem
 - 2.a) monitor the process (optional)
3. view log file (optional)
4. stop the process

Step 1 requires no further explanation.

For steps 2,3 and 5 you will need the appropriate privileges.

The name and structure of the log files that are generated depend upon the network transport that was specified when the Server Component was installed. Here, the examples are based on DEC UCX. The transport will be one of the following:

UCX	(DEC UCX)
DECNET	(DEC PATHWORKS)
TCPWARE	(Process TCPWARE)
MULTINET	(TGC Multinet)
PATHWAY	(Wollongong Pathway)

Someone in your organisation should know which one was used (the default is DECNET).

Find the Process Causing the Problem

- To see what network processes are running type:

```
$ SHOW SYSTEM/NETWORK
```

You will see a list of processes. Here is an example for DEC UCX:

```
OpenVMS V6.2 on node FULLA 27-FEB-1997 11:48:14.86 Uptime 3 01:06:40
  Pid      Process Name    State  Pri  I/O      CPU      Page flts  Pages
20200094  EVL              HIB    6   39  0 00:00:00.37    766     78
```

```

N
20200241 UCX$BOOTP_BG3 LEF 10 74 0 00:00:00.94 621 675
N
2020024F UCX$REXEC_BG298 HIB 5 406 0 00:00:06.27 2446 2021
N
    
```

- You must make a guess at which of these processes might be the offender. Look at the details of the process, to verify that it is the one causing problems. To do this you would type:

```
$ SHOW PROC/ID=<process id>/CONTINUOUS
```

<process id> is the process identifier, and is found in the first column of the output above, under the Pid heading. /CONTINUOUS instructs the output to be shown on screen.

Continuing the example, and assuming that you thought that the last process in the list was the likely offender, you would type:

```
$ SHOW PROC/ID=2020024F/CONTINUOUS
```

```

                                Process UCX$REXEC_BG298                                11:49:19
State                            HIB                            Working set                            2021
Cur/base priority                6/4                            Virtual pages                            7124
Current PC                        7FFEDF8A                            CPU time                                000:00:00:06.27
Current PSL                       03C00000                            Direct I/O                               105
Current user SP                   7FEC75D4                            Buffered I/O                             301
PID                               2020024F                            Page faults                             2446
UIC                               [MIKEU]                            Event flags                             60000001
                                                80000000

DKA300:[VERSION.RMS12B350.SYSTEM]SQLSRV.EXE;12
    
```

- If the user with a UIC (user identification code - the login name) of MIKEU was the user who sent the query from the PC, and assuming that this user does not have any other processes running on the server, then we know that this is the offending process.

Transport		Process name
UCX	(DEC UCX)	UCX\$REXEC
DECNET	(DEC PATHWORKS)	UCX\$REXEC
TCPWARE	(Process TCPWARE)	with REXEC: REXECCD_<characters> with TCP/IP: EASYSOFT_<digit>
MULTINET	(TGC Multinet)	
PATHWAY	(Wollongong Pathway)	

View Log File

- To view the log file that is generated, type:

```
$ TYPE <log file name>
```

A list of the log file names for each of the transports is shown at the end of this section.

- Continuing with our example, to see the log file for UCX-REXEC type:

```
$ TYPE SYS$LOGIN:UCX$REXECD_STARTUP.LOG
```

```
-SYSTEM-F-LINKDISCON, network partner disconnected logical link
MIKEU          job terminated at 27-FEB-1997 11:41:37.99
```

```
Accounting information:
Buffered I/O count:          159          Peak working set size:
1326
Direct I/O count:           20          Peak page file size:
6410
Page faults:                1684        Mounted volumes:
0
Charged CPU time:           0 00:00:02.53  Elapsed time:      0
00:00:14.60
$
```

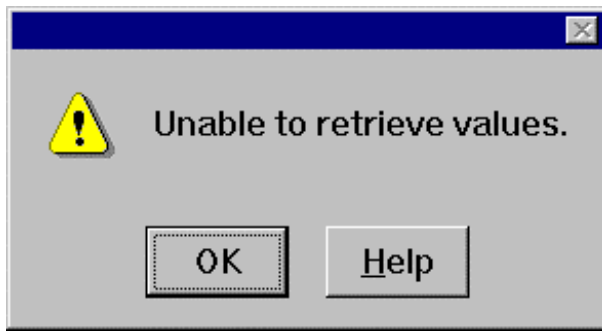
Transport	Log file name
UCX- REXEC (DEC UCX)	SYS\$LOGIN:UCX\$REXECD_STARTUP.LOG
UCX - TCP/IP (DEC UCX)	EASYSOFT_SQL_LOG:RUN_SERVER_UCX.LOG
DECNET (DEC PATHWORKS)	SYS\$LOGIN:NETSERVER.LOG
TCPWARE (Process TCPWARE)	
MULTINET (TGC Multinet)	
PATHWAY (Wollongong Pathway)	

Stop the Process

Once you know which process is causing the problem, you can stop it. To do this, type:

```
$ STOP PROCESS/IDNETIFIER=<process id>
```

When you look at the PC Application, you will see a message. What you see depends upon the application. This example is from Microsoft Excel.



APPENDIX H

Using Easysoft Support Services

These addresses/numbers are for Support. Other services are also available on different numbers.

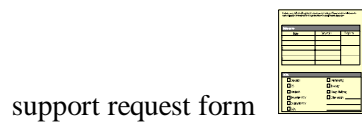


email

support@easysoft.com

fax

01937 860001



support request form

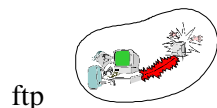
<http://www.easysoft.com/>

phone

01937 860000



mail

Thorp Arch Grange, Thorp Arch,
Wetherby LS23 7BA, United Kingdom

ftp

<ftp://ftp.easysoft.com>

Support Check List


Before contacting Easysoft, use the following check list to look for information relevant to the problem you wish to resolve. Only if you can't find anything, or if there is not enough information, contact our support team.

- Manuals (use the Index, Glossary, Contents, Lists of Figures and Tables, Troubleshooting chapters and Appendices).
- Easysoft FAQs (web site and FTP site).
- Third-party documentation where appropriate.
- Easysoft Web site - Technical Support (particularly Fixes), Help, Search.
- If you need to send data to Easysoft, refer to the next but one section, "Sending Data to Easysoft", for information on how to do this.

The Easysoft FTP Site

You've already seen the structure of the FTP site - recall the web tour - so we'll dive straight into accessing data. Let's say you want to access a FAQ sheet on Microsoft Access. There are many different implementations of FTP - here, we'll use Microsoft FTP.



1. Start a DOS session - click the MS DOS Command Prompt () in the Program Manager. The DOS prompt appears. Note the name of the current directory - this is where the file you download will be put.

2. Start an FTP session and connect to the Easysoft FTP site

```
\users\default> ftp ftp.easyssoft.com
```

3. An introductory message appears, after which you must enter a user name and password. The username to use is "anonymous" and the password is your full email address

```
User (lodur.easyssoft.com:(none)): anonymous
  331 Guest login ok, send your complete e-mail address as password.
  Password: <enter your email address here>
  <additional messages may appear here>
  230 Guest login ok, access restrictions apply.
```

4. Change directory to /pub/faq/windows and list the files there

```
ftp> cd /pub/faq/windows
```

```
ftp> dir
```

```
PORT command successful.
  150 Opening ASCII mode data connection for /bin/ls. total 19288
  <directory listing>
  <informational messages>
```

5. ftp> bin
200 Type set to I.

The files should be downloaded in binary mode.

6. Download the Access 2.0 FAQ into the current directory.

```
ftp> get access2.doc (goes in directory from which FTP was called)
```

Note that filenames are case sensitive.

7. Just to check that you have what you want, start Microsoft Word, and look at this file.

Sending Data to Easysoft

To send data to Easysoft do the following:

1. Preparation

If you are requested to send your data to Easysoft, you need to make sure that you send both the Easysoft Catalog and the data. Before you back up the data make sure that no one is using the software by using the Host Administrator SHOW USERS command:

```
$ RUN EASYSOFT_SQL_ADMIN
```

```
ADMIN> SHOW USERS
```

2. Sending data

- On tape

Back up the Catalog and the data. You can back up to TK50 or DAT and send it in the post. e.g.

```
$ BACKUP/REWIND/VERIFY EASYSOFT_SQL_CATALOG:*.*,DATA:*. * -
MKA700:EASYSOFT/LOG
```

- On floppy

Back up the Catalog and the data to a disk file. e.g.

```
$ BACKUP/REWIND/VERIFY EASYSOFT_SQL_CATALOG:*.*,DATA:*. * -
EASYSOFT.BCK/SAVE/LOG
```

Create a Zip file using ZIP. e.g.

```
$ ZIP:==$EASYSOFT_SQL_SYSTEM:ZIP.EXE
$ ZIP "-v" EASYSOFT.ZIP EASYSOFT.BCK
```

Send the zipped files on a floppy disk to Easysoft.

- Electronically

Create a .ZIP file (see above) and then use one of the following methods:

Internet Mail: support@easysoft.com

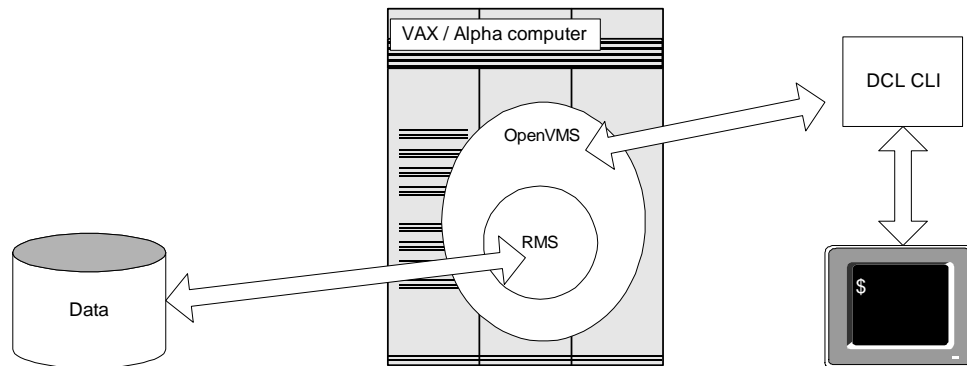
Internet FTP: ftp.easysoft.com

APPENDIX I

RMS in Context

This appendix presents RMS within the context of OpenVMS and DCL.

Relationship between the components



Digital Equipment Corporation (DEC) is the company which produces VAX and Alpha computers. *OpenVMS* (formerly VMS, Virtual Memory System) is a general-purpose, multi-user operating system which is used by VAX and Alpha computers.

OpenVMS Alpha supports Digital Equipment Corporation's Alpha series computers, and OpenVMS VAX supports VAX, MicroVAX, VAXstation, and VAXserver series computers. For convenience, we use the term OpenVMS to include both OpenVMS Alpha and OpenVMS VAX.

Aside - VAX and Alpha Architecture

VAX (Virtual Address Extension) is a high-performance, multiprogramming computer system based on a 32-bit architecture. The VAX architecture is a robust, flexible, complex instruction set computer (CISC) architecture used across the entire family of VAX systems.

The Alpha architecture newer and faster than VAX. It is a high-performance reduced instruction set computing (RISC) architecture that can provide 64-bit processing on a single chip. It processes 64-bit virtual and physical addresses and 64-bit integers and floating-point numbers. The 64-bit capability is especially useful for applications that require high-performance and very large addressing capacity.

DCL (DIGITAL Command Language) is the standard user interface to all OpenVMS and VMS systems. A user writes DCL commands to access the routines provided by the OpenVMS operating system.

RMS is one of the data management facilities provided by OpenVMS. It interprets the record structure of files and it provides a set of routines which is used to manipulate files. Specifically, this manipulation is the opening, closing, reading from, writing to, extension of and deletion of files.

RMS promotes safe and efficient *file sharing* by providing multiple file access modes, automatic *record locking* (where applicable), and optional *buffer sharing* by multiple processes.

RMS utilities and a File Definition Language (FDL) aid file creation and record maintenance. The utilities convert files from one organisation and format to another; restructure indexed files for storage and access efficiency; and reclaim data structures within indexed files. The FDL is a special purpose language that is used to describe the file organisations for data files.

GLOSSARY

Buffer A temporary storage space for data.

FDL (File Definition Language) A language used to describe file organisations of data files.

File organisation The file structure that is used as the physical arrangement of the data on the storage medium.

Operating System An integrated collection of programs that performs system functions.

Record locking The ability to control the operations being performed on a file that is being simultaneously accessed by more than one program. It prevents two programs accessing the same record.

Routine A set of computer instructions that perform an operation.

Overview of OpenVMS

This section reviews the basic concepts of OpenVMS. For the exercises in the modules you should have a working ability with the material presented here, but you do not need to be an expert.

The OpenVMS operating system includes a number of services:

- **DCL.** This provides a consistent interface with which to access the OpenVMS operating system. It is discussed in more detail in “Digital Command Language”, appendix page 56.
- **RMS.** This provides an interface at the application-program level to record and file management functions.
- **Runtime Library.** This is a set of language-independent procedures that establish a common runtime environment for application programs written in any VAX language. Application programs can be composed of modules written in many different programming languages. The runtime library is not discussed further.
- **OpenVMS System Services.** These are procedures provided by the operating system to control resources available to processes, provide for communication among processes and perform basic operating system functions such as I/O coordination. They are not discussed further.

When you log in to OpenVMS from a terminal an interactive OpenVMS session is started. An *interactive process* is created which performs work for you. This process executes the programs that make up the applications that you run on the computer.

After you log in, DCL checks your login directory (defined later) for a file called (by default) LOGIN.COM. If it exists, it is automatically invoked and all the commands contained in it are executed before DCL issues the first command prompt (defined later). The login procedure can be used to set a personal working environment, for example, to change the DCL prompt or to define logical names (defined later).

Once an interactive session has started you are able to tell OpenVMS what to do. This is accomplished by issuing a *command* at the terminal. This command will be carried out by a special program called a *command language interpreter* (CLI). In OpenVMS the standard CLI is called DCL. DCL takes commands from the keyboard, analyses them and then performs them.

During an interactive session control of the terminal lies with DCL. To indicate that it is ready to accept a command, it display the *DCL prompt*. This is a dollar sign (\$) by default, but you can change it if you wish. All DCL commands conform to the same basic format; this is described in more detail in “Digital Command Language”, Appendix page 56.

All or part of a file specification (a unique identification for a file - see “File Specifications in OpenVMS”, Appendix page 54) can be replaced by a *logical name* (or *logical*). The

main purpose of logicals is to relieve the user from having to remember the disk and directory location of a file.

Objects such as files and disk volumes can be protected so that only certain users can perform specified operations. For example, a file can be protected so that all users can read it, but only the creator can modify or delete it. This object protection uses the *user identification code* (UIC), which is a number assigned to objects to identify the *owner* of the object. A UIC is assigned to each user. When you log in to OpenVMS, your UIC is located, and becomes associated with the interactive process that OpenVMS creates for you.

Privileges are used to restrict the use of system functions and resources which could have a negative impact on the system when they are used indiscriminately. For example, the ability to delete files that normally cannot be deleted. *Authorised privileges* are assigned to a user by the system manager. These are the privileges that the user is allowed to enable. *Process privileges* are the privileges that the user has actually enabled. The system manager determines which privileges will be enabled when a user logs in.

GLOSSARY

Command An instruction (usually an English word) that specifies an operation to be performed.

DCL prompt By default this is the dollar sign (\$). It indicates that DCL is ready to accept a command.

File specification A unique identification for a file which gives its physical location, the file type and the version number.

Logical (name) An alias for a file specification.

Privilege A characteristic which specifies the allowed operations that a user or program can perform.

File Specifications in OpenVMS

One of the purposes of an operating system is to support the creation and manipulation of data files. In OpenVMS a file is designated by a sequence of characters called a *file specification* (or *file spec*). The complete format for a file specification is:

```
node::device:[directory...]name.type;version
```

Node identifies a particular node in a DECnet network. The name of the node is followed by two colons, to distinguish it from the device name. The node can be omitted, in which case the local node is assumed.

Device specifies the disk or tape on which the file resides. It can be omitted, in which case the device containing the working directory is assumed. (The working directory is the

directory in which a user is currently working, and is established with the **SET DEFAULT** command.)

Directory specifies directory location of the file. If this is omitted, the working directory is assumed.

Name, in conjunction with the type component, identifies the file in the selected directory.

Type is used along with the file name to uniquely identify a file in a directory.

Version specifies the version number of the file. In OpenVMS (unlike, for example, DOS) a given file that is specified by a name and type combination can have multiple copies in the same directory. These copies are accessed by specifying the version number. If version number is omitted, then the latest version is assumed.

When a user logs in to OpenVMS the *login directory*, also known as *home directory*, is established. At any time throughout an OpenVMS session one directory is chosen as the *default directory*. When a file is accessed with a file specification that does not include a device and directory, the file is assumed to reside in the default directory. When a user logs in, the login directory is automatically established as the default directory. The default directory can be changed with the **SET DEFAULT** command.

Wildcards are special characters that can be inserted into a normal file specification. A file specification can include the following wildcard characters:

- * The asterisk character matches zero or more characters. It can be used in a directory name, file name, type, or version.
- % The percent sign matches exactly one character. It can be used in the directory name, file name or type.
- ... An ellipsis matches the entire directory subtree below the named directory. It can be appended to a directory name in a directory specification.

Digital Command Language

This section is provided for the benefit of people who are not familiar with DCL. It is by no means comprehensive, its function is purely to provide a basic reference so that you will be able to perform the exercises in the modules.



DCL Commands

The structure of DCL commands conform to the same basic format, which is a command verb followed by parameters and qualifiers if appropriate.

The *command verb* is one or more words that identifies the command. Command verbs can be abbreviated provided the abbreviation is unambiguous. For example, the COPY command can be abbreviated to COP, but not to CO, because the latter could not distinguish between COPY, CONVERT and CONTINUE. A command verb is guaranteed to be unique if four or more letters are specified. In these examples, the full command is shown.

Some commands take one or more *parameters* after the verb. A parameter is an item of information that is used to refine the action that will be taken by the command. The verb specifies the action and the parameter specifies the object to be acted upon.

In some cases parameters are always needed, in which case they are called *required parameters*. For example, the DELETE command always needs a file specification as a parameter e.g.

\$ **DELETE SALES.DAT;* _____** *using the wildcard character (*) results in the deletion of all versions of SALES.DAT*

Optional parameters are specified only when they are needed. For example, to see a list of all logicals, you would (in full) type:

\$ **SHOW LOGICAL**

However, if you wanted to see just the Easysoft logicals, then you would use an optional parameter (which specifies those logicals that you want to see) e.g.

\$ **SHOW LOGICAL EASY***

DCL commands can be further modified by the use of command *qualifiers*. Most qualifiers are optional, and when specified they alter some aspect of the operation that is performed by the command. Qualifiers consist of words preceded by the forward slash character (/). Sometimes, qualifiers require a value. In this case, the qualifier name is followed by an equal sign (=) and then the value. For example, to delete all the old versions of the SALES.DAT file created before the 1st of February 1996 you would type:


```
$ DELETE /BEFORE=1-FEB-1988 SALES.DAT;*
```

Commands can be split over multiple lines by ending each line (except the last) with a hyphen (-). A command can be split wherever a space or a comma appears.

Command Procedure

A sequence of commands can be grouped and saved in a text file. Later, this sequence of commands can be executed. The file in which the commands are saved is called a *DCL command procedure*.

Each command in the procedure must start with a dollar sign, which indicates to DCL that the line contains a command which it should interpret. There can be one or more spaces after the dollar sign. Then the DCL command is specified exactly as it would be typed at the DCL prompt.

DCL command procedure files are given the file type COM (for command) by convention. Once a procedure file has been created, it can be played back using the at-sign (@) command. DCL assumes that the file type of a procedure is COM, so this can be omitted (if the type really is COM).

The text file that you create can be made visually more appealing by separating commands with one or more lines containing just the \$ sign (lines cannot be completely blank). Comments can be included on a line. The comment indicator is the exclamation mark (!); this, and anything after it will be ignored by DCL. If you want a comment to extend over two or more lines, you must put an exclamation mark on each line.

GLOSSARY

Command An instruction (usually an English word) that specifies an operation to be performed.

Parameter A value that is passed to a command.

Qualifier A portion of a command string that modifies a command verb or a command parameter. It has the format: /qualifier[=option].

Global Index

The page numbering shows the module number followed by the page number within the module.

special characters

% in SQL LIKE operator, 2-14
* (in SQL statement), 2-9

A

Access control list. *See* ACL
Access mode. *See* Record access mode
Access rights of an object. *See* Privileges under *object*
ACL, 3-22
ACL-based file protection. *See* File protection
Active file
 view, 4-14
Add Criteria dialog box (MS), 6-5
Add criteria in application. *See under specific application*
Add Data Source dialog box (MS), 5-3
Add rows to table. *See* INSERT statement (SQL)
Add Tables dialog box (MS), 4-10, 6-4
Add-Ins dialog box (MS Excel), 12-3, 12-4
ADMIN user
 password. *See under* Passwords and usernames
Administrator greetings screen, 8-5
Aggregate functions, 2-16
Allow editing in MS Query. *See* Microsoft Query
Alpha
 architecture, A-53
Alternate key (RMS). *See under* Key (RMS)
American National Standards Institute. *See* ANSI
Analyze/RMS_File utility, 7-5
AND operator (SQL), 2-11
ANSI, 1-13
API, 1-4
 conformance. *See* ODBC conformance
Application
 using ODBC, 1-5
Application Programming Interface. *See* API
Approximate Numeric Data Types. *See under* Data types
Architecture
 Easysoft. *See* Easysoft architecture
 ODBC. *See under* ODBC
Ascending index (SQL). *See* Index (SQL)
Available Data Sources list box, 6-2
AVG function (SQL), 2-16

B

BEGIN STRUCTURE element, A-35
BETWEEN operator (SQL), 2-13
Block I/O, 3-16
Brackets. *See* parentheses
Brackets in SQL. *See under* SQL, parentheses
BROCHURE table, 11-13
BROCHURE.DAT, 10-3
BROCHURE.FDL, 10-12
Brochures
 list of, 10-7
Byte order
 reversed, 3-28

C

C:\EASYSOFT\SQL\SYSTEM, 8-23, 8-32
Catalog. *See* Easysoft Catalog
 CSV description, 10-21
 functions, 1-10
 importing to server, 9-5
 Impromptu, 6-24
Catalog administrator
 password. *See under* Passwords and usernames
Catalog Driver, 4-4
Catalog Login
 changing password, 9-3
Catalog tables
 view, 4-10
Catuser routine, 9-3
Cell reference, 12-20
Change rows in table. *See* UPDATE statement (SQL)
Character String Data Types. *See under* Data types
Characteristics of a file. *See* File attributes
Characters, valid for definitions. *See under appropriate definition*
Characters, valid for names. *See under appropriate name*
Choose Field dialog box (Crystal Reports), 6-33
Choose Fields dialog box (Lotus), 6-18
Choose SQL Table dialog box (Crystal Reports), 6-31
COBOL converter
 abort conversion process, A-23
 conformance, A-21

Database Name, 9-8, A-24
 Default Directory, 9-8, A-24
 FILLER fields, A-25
 hierarchical field structure, A-25
 hyphens - conversion to underscore, 9-8, A-24
 hyphens - convert to underscore, A-24
 installation, A-21
 invalid syntax in COBOL file, A-23
 maximum size of COBOL file, A-23
 purpose, A-21
 removing, A-27
 size of installed files (RMS), A-21
 view output on screen, A-23
 Cognos Impromptu. *See* Impromptu
 Collation
 ascertaining, 7-29
 Column definition, 4-6
 Column Definitions dialog box (ES), 8-16
 Column heading
 changing, 11-3
 Column name
 allowed characters, 2-8
 Column Selection dialog box (RMS macro), 12-7
 Columns
 defined, 2-3
 defining, 8-16
 Combining conditions in SQL, 2-11
 Combining data from different tables. *See* Join
 Command procedure (DCL), A-59
 Command verb (DCL)
 defined, A-58
 Company
 example for training, 10-1
 Comparison operator. *See* Scalar comparison operator (SQL)
 Condition variable (PDL), A-37
 Conditions
 combining in SQL. *See* Combining conditions
 Connect Timeout. *See* Timeouts
 Connect to External dialog box (Lotus), 6-16
 CONTACTS table (local Access), 11-43
 CONVERT command (RMS), 7-20
 Converting data types. *See under* Data types
 Copying definitions. *See under appropriate definition*
 COUNT (*) function (SQL), 2-16
 COUNT DISTINCT function (SQL), 2-16
 Create New Data Source dialog box (MS), 5-9
 Create Report Expert dialog box (Crystal Reports), 6-31
 Criteria
 parentheses, 11-12
 Criteria pane, 6-4
 Criteria Selection dialog box (RMS macro), 12-7, 12-12
 Crystal Reports, 6-30
 add criteria, 6-33
 view SQL, 6-34
 CTL3D.DLL, A-41
 Currencies
 list of, 10-8
 Currency code, 10-3

Currency conversion, 11-18
 Currency rates
 obtaining from Internet, 11-18
 CURRENCY table, 11-21
 CURRENCY_MASTER table, 11-17
 CURRENCY_MASTER.DAT, 8-2, 8-17, 9-11, 10-3
 CURRENCY_MASTER.FDL, 10-13
 CURRENCY_RATE table, 11-17
 CURRENCY_RATE.DAT, 3-25, 7-4, 7-5, 8-2, 9-11, 10-3
 CURRENCY_RATE.FDL, 7-8, 10-13
 CUSTOMER table, 11-11, 11-23, 11-25, 12-11
 CUSTOMER.DAT, 10-4
 CUSTOMER.FDL, 10-14
 Customers
 list of, 10-6
 Customised reports. *See under* RMS macro

D

Data
 sending to Easysoft, A-50, A-52
 Data Output dialog box (RMS macro), 12-9
 Data source, 1-4, 1-11
 configuring, 5-10
 connecting to, 2-32
 defined, 1-6
 method of definition, 4-5
 naming restrictions, 5-4
 removing, 5-10
 setting up, 5-2, 5-9
 settings, 5-6
 validation of, 5-5
 Data source name. *See* DSN
 Data Sources dialog box (16 bit), A-19
 Data Sources dialog box (MS), 5-3
 Data types
 Approximate Numeric, A-10
 Character String, A-10
 conversion, 1-11
 Datetime, A-10
 Exact Numeric, A-10
 purpose of, 2-14
 SQL, A-10
 supported by Easysoft SQL, A-11–A-14
 Database Definition dialog box (Impromptu), 6-26
 Database Definitions dialog box (ES), 8-13
 Database dialog box (MS), 6-11, 6-12
 Database management system. *See* DBMS
 Database Security dialog box (ES), 8-18
 Database structure
 pictorial representation, 2-8
 Databases
 defined, 1-4, 2-3, 4-5
 defining, 8-13
 setting privileges, 8-18, A-27, A-37
 specifying in data sources, 5-7
 Databases dialog box (Impromptu), 6-25
 DATE data type, 1-10

- Dates
 - defining, 8-12
 - formats, 8-12
 - returned in wrong order, 3-28
 - stored as ASCII, 3-28
 - Datetime Data Types. *See under* Data types
 - DBMS, 1-4
 - DCL, A-55
 - defined, A-54
 - Default directory
 - defined, A-57
 - Defining an object. *See under appropriate object*
 - Definitions
 - exporting, 8-23
 - importing and exporting, 8-23–8-26
 - Definitions, imported
 - changing, 8-25, 8-26
 - DELETE statement (SQL), 2-20
 - Deleting definitions. *See under appropriate definition*
 - Descending index (SQL). *See* Index (SQL)
 - DESTINATION table, 11-10, 11-21
 - DESTINATION.DAT, 10-4
 - DESTINATION.FDL, 10-15
 - Destinations
 - list of, 10-9
 - Digital Command Language. *See* DCL
 - DIRECTORY command (RMS), 7-4
 - Disable Winsock. *See under* Winsock
 - Download System Catalog dialog box (ES), 8-8
 - Download Wizard (RMS macro), 12-7
 - Driver. *See* ODBC driver
 - Driver Manager, 1-5, 1-6, 4-15
 - DSN, 1-11
 - Duplicate rows
 - eliminating, 2-15
 - Dynamic tracing. *See* Function calls, tracing
-
- ## E
-
- Easysoft Administrator
 - access security, 8-6
 - invalid characters, A-36
 - licence information, 8-5
 - Easysoft Administrator - Log on dialog box, 8-6
 - Easysoft Administrator dialog box, 8-7
 - Easysoft Administrator files
 - default location, 8-32
 - Easysoft Administrator installation, 8-31
 - Easysoft Administrator Setup (Directory) dialog box, 8-32
 - Easysoft Administrator Setup (Introductory) dialog box, 8-32
 - Easysoft Administrator Setup (Successful Installation) dialog box, 8-33
 - Easysoft Administrator Setup (User Details) dialog box, 8-32
 - Easysoft architecture, 4-3
 - Easysoft Catalog, 2-28, 4-4
 - creating, 9-2
 - directory, 9-2
 - downloading, 8-8
 - importing definitions to local copy, 8-25, 8-26
 - local copy, 8-6
 - number of, 4-9
 - password. *See under* Passwords and usernames
 - purpose of, 4-9
 - specifying location of, 5-4
 - update precautions, 8-8
 - uploading, 8-19
 - user information, 4-9
 - username. *See under* Passwords and usernames
 - Easysoft Client Component, 4-3
 - Easysoft COBOL Converter. *See* COBOL converter
 - Easysoft Excel Macro for RMS. *See* RMS macro
 - Easysoft ODBC
 - cancelling installation, A-19
 - driver installation, A-16
 - driver versions, A-16
 - installation, A-17
 - memory requirement, A-16
 - storage requirement, A-16
 - system requirement, A-16
 - upgrading, A-16
 - Easysoft ODBC driver, 4-15
 - Easysoft ODBC Logging dialog box, A-39
 - Easysoft ODBC login prompt, 8-9
 - hiding and showing, 5-6
 - Easysoft ODBC Settings dialog box, 5-6
 - Easysoft ODBC Setup dialog box, 5-4, 9-6, A-19
 - Easysoft PowerHouse PDL Converter. *See* PDL Converter
 - Easysoft Query for Windows
 - Message window, defined, 2-32
 - Output window, defined, 2-32
 - Query Wizard, 2-34
 - SQL window, defined, 2-32
 - Easysoft Server Component, 4-3
 - Easysoft SQL, 4-3
 - Easysoft Support
 - check list, A-50
 - contact options, A-49
 - Easysoft Travel Company, 10-1
 - EASYSOFT.INI, A-40
 - EASYSOFT_SQL_CATALOG, 7-3
 - EASYSOFT_SQL_COBOL_CATALOG, 9-11
 - EASYSOFT_SQL_COBOL_DATA, 9-11
 - EASYSOFT_SQL_DATA, 7-3, 9-11
 - EASYSOFT_SQL_FDL, 7-3
 - Easysql, 9-7
 - Edit/FDL utility, 7-9
 - Elementary items (COBOL), A-25
 - Eliminating duplicate rows. *See under* Duplicate rows
 - END STRUCTURE element, A-35
 - Error messages, 5-12
 - Errors
 - dealing with in ODBC. *See under* ODBC
 - ESNET.LOG, A-40
 - ESODBC.LOG, A-40

Exact Numeric Data Types. *See under* Data types
 Exchange rate, 10-3
 Export File Definitions dialog box (ES), 8-23

F

FDL, 7-2
 structure of, 7-26
 FDL editor. *See* Edit/FDL utility
 FDL file, 7-9
 Field Details dialog box (ES), 8-10
 Fields
 defining, 8-10
 mapping to columns, 8-14
 File data source, 5-8
 File definition, 4-6, 8-9. *See also* Files, defining
 creating new, 8-9
 displaying current, 8-9
 reasons for replication, 4-7
 File Definition dialog box (ES), 8-10
 File Definition Language. *See* FDL
 File organisation
 compared with each other, 3-4
 compared with file type, 3-4
 determining, 7-4
 File protection, 3-20
 File specification, 4-6
 File structure
 defining part of, 4-7, 4-8
 File type
 compared with file organisation, 3-4
 Files
 accessed by a query, A-39
 attributes, 3-5
 creating from FDL, 7-18
 defining, 8-7–8-22
 indexed, 3-9, 3-12
 mapping to SQL tables, 4-6–4-8
 pre-requisites to defining, 8-3
 relative, 3-7
 sequential, 3-6
 stages of defining, 8-3
 FILLER fields (COBOL). *See under* COBOL
 converter
 FILLER fields (PDL). *See under* PDL converter
 Fixed-length record format. *See* Record format
 Formula bar, 12-20
 Function call, 1-6
 Function calls
 tracing, A-42

G

Get External Data dialog box (MS), 6-8
 Group items (COBOL), A-25

H

Header and detail records

defining in SQL terms, 4-8
 HEADER table, 11-28, 11-31, 11-33, 11-35, 11-37
 Hierarchical field structure (COBOL). *See under*
 COBOL converter
 Home directory. *See* Login directory
 Host Administrator, 4-4

I

I/Os per query
 number of, A-39
 Import New File Definition dialog box (ES), 8-24
 Impromptu, 6-24
 add criteria, 6-28
 view SQL, 6-29
 Index
 compared to RMS key, 3-14
 Index (SQL), 2-5
 Indexed file organisation. *See* File, indexed
 Indexed Sequential Access Method, 3-16
 Initial Password dialog box (ES), 8-6
 Inner join. *See under* Join
 INSERT statement (SQL), 2-18
 Install Drivers dialog box, A-18
 Installing Easysoft Administrator. *See under*
 Easysoft Administrator installation
 International Standards Organisation. *See* ISO
 Invoice, 10-4
 INVOICE.DAT, 10-4
 INVOICE.FDL, 10-15
 ISAM. *See* Indexed Sequential Access Method
 ISO, 1-13

J

Join, 2-6
 in SQL, 2-14
 inner, 2-6
 outer, 2-6
 types, 2-6

K

Key (RMS), 3-9
 adding to file, 7-19
 alternate, 3-11
 compared to index, 3-14
 determining, 7-8
 primary, 3-11
 segmented, 3-10
 simple, 3-9
 Key field. *See* Key (RMS)

L

Left outer join. *See under* Join

LIKE operator (SQL), 2-13, 2-14
 Link dialog box (MS), 6-10
 Link Tables dialog box (MS), 4-12, 6-11
 Log On Server dialog box (Crystal Reports), 6-30
 Logical (name)
 defined, A-55
 Login directory, A-57
 LOGIN.COM, A-55
 Lotus 1-2-3
 add criteria, 6-18
 using, 6-15
 view SQL, 6-19
 lotus.bcf file, 6-15

M

Mail Merge, 6-20
 Mail Merge Helper dialog box (MS), 6-20
 Mail Merge toolbar, 6-22
 MAX function (SQL), 2-16
 Messages
 hiding, A-40
 Microsoft Access
 add criteria, 6-12
 design mode of table, 6-14
 using, 6-10
 view indexes, 6-14
 view SQL, 6-14
 wildcard character, 11-3
 Microsoft Excel
 Formula bar. *See* Formula bar
 reconnection to data source, 11-3
 show column headings, 6-8
 show row numbers, 6-9
 specify download location, 6-9
 using, 6-2
 Microsoft Internet Explorer, 11-18
 Microsoft ODBC Administrator, 5-1, A-19
 apparent version anomaly, A-16
 purpose of, 5-2
 version 2.5 (16 bit), 5-2
 version 2.5 (32 bit), 5-2
 version 3.0, 5-2, 5-8
 version shipped, A-16
 version shipped with Easysoft, 5-2
 Microsoft ODBC Setup
 running, A-17
 Microsoft Query
 add criteria, 6-5, 6-7
 allow editing, 11-3
 modify SQL in query, 6-7
 reading file, 11-3
 reconnection to data source, 11-3
 using, 6-2
 view SQL, 6-7
 wildcard character, 11-3
 Microsoft Word Mail Merge. *See* Mail Merge
 MIN function (SQL), 2-16, 11-30

Modify rows in table. *See* UPDATE statement (SQL)
 Multiple defined fields (PDL)
 prevention of multiple views, A-37
 Multiple definitions (COBOL)
 prevention of multiple views, A-27

N

Naming restrictions in data source. *See under* Data source
 Network connection
 defining packet size. *See* Packet size
 packet acknowledgement. *See* Packet acknowledgement
 Network statistics, A-39
 Network transport
 PC options, 5-4
 New Catalog dialog box (Impromptu), 6-25
 New Connection dialog box, 2-32
 New dialog box (Impromptu), 6-27
 New Query Assistant dialog box (Lotus), 6-16
 New Query dialog box (MS), 6-12
 Non-unique index (SQL). *See* Index (SQL)
 NOT operator (SQL), 2-11
 NULL operator (SQL), 2-13

O

Object privileges. *See under appropriate object*
 Object protection, A-56
 Object security. *See* 'privileges' under *appropriate object*
 Objects dialog box (ES), 8-7
 ODBC
 architecture, 1-5
 components
 application, 1-5
 data source, 1-5
 determining details of, 5-10
 driver, 1-5
 driver manager, 1-5
 conformance, 1-13
 error checking, 1-6
 purpose of, 1-4
 ODBC Administrator icon, A-19
 ODBC call, 4-15
 ODBC Data Sources dialog box (MS), 6-3
 ODBC date format, 2-17
 ODBC driver
 16 and 32 bit. *See under* Easysoft ODBC driver
 functions of, 1-8
 information about, 5-10
 listing, 5-10
 multi-tier. *See* two-, and three-tier
 single-tier, 1-6
 three-tier, 1-6
 two-tier, 1-6
 ODBC Drivers tab, 5-10

ODBC icons, A-20
 ODBC Options dialog box (MS), A-42
 ODBC Setup (welcome) dialog box, A-18
 ODBC.INI, 4-15
 Olsen and Associates Currency Converter, 11-18
 One-to-many relationship, 2-8
 Open Data Source dialog box (MS), 6-21
 OpenVMS
 defined, A-53
 Options dialog box (MS), 4-12
 Options dialog box (RMS macro), 12-5
 OR operator (SQL), 2-11
 ORDER BY statement (SQL), 2-15
 Order data
 relational. *See* Relational data
 Ordered retrieval of data. *See under* Relational data
 Outer join. *See under* Join
 Output Destination dialog box (RMS macro), 12-8, 12-13

P

Packet acknowledgement
 enabling, 5-7
 Packet size
 defining, 5-6
 Parameter (DCL), A-58
 Parentheses
 using in SQL, 2-11
 Passwords
 exporting, 8-24, 9-4
 Passwords and usernames
 ADMIN user, 9-2
 catalog administrator, 9-2
 Easysoft Administrator
 creating password, 8-5
 creating username, 8-32
 for Easysoft Catalog, 5-5
 Server, 5-5
 PDL Converter
 abort conversion process, A-32
 Database Name, A-33
 Default Directory, A-33
 FILLER fields, A-34
 FILLER-NUMERIC fields, A-34
 hyphens - conversion to underscore, A-34
 invalid characters in Easysoft Administrator, A-36
 invalid syntax in PDL file, A-32
 maximum size of PDL file, A-32
 purpose, A-31
 removing, A-37
 sub-structured fields, A-35
 view output on screen, A-32
 Physical characteristics of a file. *See* File attributes
 Pivot Output (data) dialog box (RMS macro), 12-14
 Pivot Output dialog box (RMS macro), 12-14, 12-15

Pivot Table, 12-13
 Precedence order, 2-11
 Preview Sample dialog box (Crystal Reports), 6-32
 Primary key (RMS). *See under* Key (RMS)
 Privileges, A-56
 Privileges of an object. *See under appropriate object*
 Prolog structure, 3-13
 Protection code, 3-21
 Protection mask. *See* Protection code
 Purchase invoice, 10-4, 11-28, 11-33, 11-37

Q

QSHOW converter. *See* PDL Converter
 Qualifier (DCL), A-58
 Query and Pivot toolbar (MS Excel), 12-15
 Query dialog box (Impromptu), 6-27, 6-28
 Query Execution Plan, A-39

R

Random record access mode. *See under* Record access mode
 Receive Timeout. *See* Timeouts
 Record
 determining fields in, 7-7
 field length, 8-2
 offset, 8-2
 variable length, 3-8
 Record access mode, 3-15
 Record file address, 3-15
 Record format, 3-17
 Record Management Services. *See* RMS. *See* RMS
 Record Size field
 default value, 8-9
 Record structure
 defining, 4-6
 REDEFINES keyword (COBOL), A-27
 REDEFINES keyword (PDL), A-37
 Registry, 4-15
 Relational data
 date format, 2-4
 ordered retrieval, 2-5, 2-15
 structure, 2-3
 Relative file organisation. *See* File, relative
 Relative record number, 3-7, 3-15
 Remote Command
 specifying name of, 5-4
 Remote Object
 specifying name of, 5-4
 Remote Service
 specifying name of, 5-4
 Remove rows from table. *See* DELETE statement (SQL)
 Removing definitions. *See under appropriate definition*

Removing duplicate rows. *See under* Duplicate rows, eliminating

RENAMES keyword (COBOL), A-27

Reserved words (SQL), 2-29

retrieving RMS data. *See* RMS data

Reversed byte order. *See* Byte order

RFA. *See* Record File Address

Right outer join. *See under* Join

RMS, A-55

- defined, A-54
- function of, 3-3

RMS data

- making relational, 4-3
- retrieving - expected results, 11-2
- retrieving and updating, 11-1–11-45

RMS Driver, 4-4

RMS macro

- contact information, 12-2
- customised reports, 12-18
 - creating, 12-18
 - generalise a report, 12-20
 - running, 12-19
- debug options, 12-6
- dialog boxes. *See under name of box*
- download data, 12-7
- failed update colour, 12-16
- filename, 12-2
- font colour after upload, 12-16
- initialisation, 12-5
- installation, 12-3
- keeping connection open, 12-5
- location of add-in software, 12-3
- maximise access rate, 12-5
- multiple queries on worksheet, 12-22
- number of rows returned, 12-5
- on-screen error messages, 12-6, 12-17
- overview, 12-2
- refresh data, 12-9
- search criteria, 12-7, 12-11
- select output worksheet, 12-8
- set worksheet column widths, 12-9
- setting timeout value, 12-5
- sort data, 12-8
- status message, 12-6
- status of upload, 12-16
- successful insert colour, 12-16
- successful update colour, 12-16
- update sequence rules, 12-16
- upgrading, 12-4
- upload, 12-16
- validation of SQL, 12-10
- version information, 12-2
- worksheet column headings, 12-9

Rms menu option, 12-2

RMS Utilities, 4-4

Rounding in SQL, 2-19

Rows

- defined, 2-3

Runtime Library, A-55

S

Sales invoice, 10-4, 11-30, 11-33

Saving changes to an object. *See under appropriate object*

Scalar comparison operator (SQL), 2-10

Select Data Source dialog box, 6-3

Select Data Source dialog box (MS), 6-10

SELECT DISTINCT (SQL), 2-15

Select ODBC Trace File dialog box (MS), A-42

Select Records Expert dialog box (Crystal Reports), 6-33

SELECT statement (SQL), 2-9

Select Table dialog box (RMS macro), 12-7

Select Value(s) dialog box (MS), 6-6

Sequential file organisation. *See* File, sequential

Sequential record access mode. *See under* Record access mode

Server process

- determining, A-45
- stopping, A-47

Set Criteria dialog box (Lotus), 6-18

SHARE.EXE, A-16

Show SQL dialog box (Lotus), 6-19

Show Table dialog box (MS), 6-12

Simple key. *See under* Key (RMS)

Sort Order dialog box (RMS macro), 12-8

Special Name Characters (PDL), A-36

SQL, 1-4

- conversion in ODBC, 1-10
- defined, 2-3
- ODBC extensions, 1-10, 2-3
- purpose of, 2-3
- using parentheses, 2-11

SQL Access Group, 1-13

SQL Data Sources dialog box (MS), 8-8

SQL dialog box (MS), 6-7

SQL grammar conformance. *See* ODBC conformance

SQL query, A-39

SQL reserved words. *See* Reserved words

SQL statement, 4-16

SQL statements

- supported by Easysoft SQL, 2-28

SQLColumns function, 1-10

SQLDataSources function, 1-11

SQLDrivers function, 1-11

SQLGetFunctions function, 1-11, 1-13

SQLGetInfo function, 1-10, 1-13

SQLGetTypeInfo function, 1-11, 1-13

SQLStatistics function, 1-10

SQLTables function, 1-10

Standards for SQL. *See under* SQL

Stream record format. *See* Record format

Structured Query Language. *See* SQL. *See* SQL

Sub-structured fields (PDL). *See under* PDL converter

SUM function (SQL), 2-16

SUPPLIER table, 11-4, 11-5, 11-6, 11-7, 11-9, 11-15, 11-26, 11-27, 11-28

SUPPLIER.DAT, 10-4

SUPPLIER.FDL, 10-18
 Suppliers
 list of, 10-10
 SUPPLIERS table (local Access), 11-43
 System Catalog. *see* Easysoft Catalog
 System data source, 5-8
 System Data Sources dialog box (MS), 5-3
 System DSN tab, 5-9
 System Options (PDL), A-36

T

Table definition
 components of, 4-6
 Table Definitions dialog box (ES), 8-14
 Table name
 allowed characters, 2-8
 Table Options dialog box (MS), 4-10
 Table pane
 viewing, 6-4
 Tables
 defined, 2-3
 defining, 8-14
 differentiating between databases, 5-7
 privileges, 8-15
 Timeouts, 5-6
 Trace of Easysoft network protocols, A-40
 Trace of ODBC calls, A-40
 Tracing function calls. *See under* Function calls
 Tracing tab, A-43
 Training company. *See under* Company
 TRAINING DATA data source, 8-8, 8-20
 Transmit Timeout. *See* Timeouts

U

UIC
 defined, A-56
 UIC-based file protection. *See* File protection
 Unique index (SQL). *See* Index (SQL)
 Update rows in table. *See* UPDATE statement (SQL)
 UPDATE statement (SQL), 2-19
 Updating RMS data. *See* RMS data
 Upload dialog box (ES), 8-19
 User
 information in Easysoft catalog. *See under*
 Easysoft catalog
 User authorization file, 3-20
 User data source, 5-8
 User Definitions dialog box (ES), 8-18

User identification code. *See* UIC
 User name. *See under* Passwords and usernames
 Users
 defining, 8-17
 displaying current, 8-17
 specifying restrictions on, 8-15

V

Valid characters for definitions. *See under*
appropriate definition
 Valid characters for names. *See under*
appropriate name
 Validate Import Definitions dialog box (ES), 8-25
 Variable length record. *See under* Record
 Variable-length record format. *See* Record format
 VAX
 architecture, A-53
 VFC record format. *See* Record format
 View SQL in application. *See under specific application*
 VMS. *See* OpenVMS

W

Wildcard
 in OpenVMS, A-57
 Wildcard character
 in Microsoft Access. *See under* Microsoft Access
 in Microsoft Query. *See under* Microsoft Query
 SQL, 2-14
 Windows directory, A-40
 Winsock
 disable, A-40
 World Destinations
 list of, 10-11
 WORLD_DESTINATION table, 11-10, 11-21
 WORLD_DESTINATION.DAT, 7-10, 8-21, 9-11, 10-4
 WORLD_DESTINATION.FDL, 10-19
 creating, 7-10

X

X/Open, 1-14

