

---

# Arbor<sup>®</sup> Essbase<sup>®</sup>

Version 5

## SQL Drill-Through Guide



ARBOR<sup>®</sup>  
SOFTWARE

© 1991–1997 Arbor Software Corporation. All rights reserved.

Arbor and Essbase are registered trademarks of Arbor Software Corporation.

Microsoft is a registered trademark and Windows is a trademark of Microsoft Corporation. Excel is a product of Microsoft. Lotus, 1-2-3, and OS/2 are registered trademarks of International Business Machines Corporation. Novell and Netware are registered trademarks of Novell, Inc. All other brand and product names are trademarks or registered trademarks of their respective holders.

No portion of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or information storage and retrieval systems, for any purpose other than the purchaser's personal use, without the express written permission of Arbor Software Corporation.

**Notice:** The information contained in this document is subject to change without notice. Arbor Software Corporation shall not be liable for errors contained herein or consequential damages in connection with the furnishing, performance, or use of this material.

A R B O R   S O F T W A R E   C O R P O R A T I O N

1344 Crossman Avenue

Sunnyvale, CA 94089

Publication Number: 40-218-7300

Printed in the USA

10 9 8 7 6 5 4 3 2

**R1**

---

# Table of Contents

<b>Introduction .....</b>	<b>v</b>
About This Guide.....	v
What's Not in This Guide .....	vi
How This Guide Is Organized .....	vi
Conventions Used in This Guide.....	vii
Related Publications .....	vii
<b>Chapter 1 A Conceptual Overview .....</b>	<b>1</b>
Three-Tiered Data Architecture .....	2
The Multidimensional Model.....	2
The Relational Model.....	3
The Best of Both Worlds.....	4
The Conceptual Leap.....	5
Multidimensional Concepts .....	6
Table Concepts .....	7
The Mapping Point .....	11
<b>Chapter 2 Getting Started.....</b>	<b>13</b>
Server and Client SQL Drill-Through .....	14
SQL Drill-Through Architecture.....	15
Server SQL Drill-Through.....	15
Client SQL Drill-Through .....	16
Installing SQL Drill-Through .....	17
Installing SQL Drill-Through on a Network.....	19
Verifying an SQL Connection for Server Drill-Through .....	21

Verifying an SQL Connection for Client Drill-Through.....	23
The First Thing to Verify .....	23
Examples .....	23
The SQLDRILL.INI File .....	25
<b>Chapter 3 Defining SQL Drill-Through Mappings.....</b>	<b>27</b>
Managing the .INI Files.....	27
Designing a Profile .....	28
Understanding End-User Requirements .....	29
Sample Requirements.....	33
Requirement 1.....	33
Requirement 2.....	49
<b>Appendix A The SQLDRILL.INI File .....</b>	<b>65</b>
The Supervisor Setting.....	65
The SQLDrillServer Keyword.....	66
The Default SQLDRILL.INI File .....	66
<b>Appendix B User Exits .....</b>	<b>71</b>

---

# Introduction

The Arbor® Essbase® Multidimensional Analysis System is an on-line analytical processing (OLAP) solution that satisfies the complex calculation requirements of end-user analysts across the enterprise in various departments, including financial, accounting, and marketing. Essbase operates in a client-server computing environment on a local area network (LAN). This client-server computing environment conveniently allows multiple users to retrieve and analyze centralized data with their desktop personal computers.

The capabilities of Essbase can be extended through add-on products, such as the following:

- The ability to navigate from a view of Essbase data to a view of relational data using Arbor Essbase SQL Drill-Through and Arbor Essbase Spreadsheet Add-in.
- Arbor Essbase Currency Conversion to translate, analyze, and report on foreign financial data.
- Custom application development through the Arbor Essbase API.
- Direct access to relational data via Arbor Essbase SQL Interface.

---

## About This Guide

The *SQL Drill-Through Guide* is intended for database administrators who are responsible for providing SQL Drill-Through capabilities to end users.

This guide provides:

- Overview information on the basic concepts of multidimensional database design and its mapping to relational schema.
- Procedures for installing the SQL Drill-Through product.
- Information about how to define an SQL Drill-Through mapping through the use of sample end-user requirements.

You need to be familiar with relational table schema, SQL syntax, and the Essbase multidimensional schema.

---

## What's Not in This Guide

This guide is not intended to be used by end users to understand how the SQL Drill-Through product works. The *Spreadsheet Add-in User's Guide* has a tutorial with examples of SQL Drill-Through retrievals.

This guide does not describe how to install SQL Interface or how to define relational data sources. Procedures for installing and configuring the SQL Interface are covered in the *Installation Notes* and the *SQL Interface Guide*.

This guide also does not describe how to define and maintain Essbase applications. To use SQL Drill-Through, the Essbase database must already be defined, and you need to understand how to map the Essbase database to its relational sources. For information on designing an Essbase application, refer to the *Database Administrator's Guide*.

---

## How This Guide Is Organized

The *SQL Drill-Through Guide* is organized into the following chapters:

- Chapter 1, "A Conceptual Overview," provides a basic overview of multidimensional database concepts, and how they relate to a relational model.
- Chapter 2, "Getting Started," explains how to install the SQL Drill-Through product.
- Chapter 3, "Defining SQL Drill-Through Mappings," explains how to define the mapping between the multidimensional and relational schemas by walking through several end-user requirements.
- Appendix A, "The SQLDRILL.INI File," describes portions of the SQLDRILL.INI file.
- Appendix B, "User Exits," describes how to connect your own libraries to SQL Drill-Through.

---

## Conventions Used in This Guide

The following conventions are used throughout this guide:

- Procedures that you should follow in order are indicated with numbered steps: 1., 2., 3., and so on.
- Bullet lists, such as this one, ( • ), provide information, but not procedural steps.
- Essbase dimension names and member names appear in `This Font`. The names of files, directories, and specific text you must type appear in `This Font`. The names of dialog boxes and their controls, such as option buttons and text boxes, appear in **boldface type**.
- Titles of other books are shown in *italics*. Italics also indicate important terms and special emphasis.
- Pop-up menu items are separated from top-level menu items (those that appear in the menu bar) by a vertical bar ( | ). For example, the FlashBack command on the Essbase menu in the Spreadsheet Add-in is written as Essbase | FlashBack.

---

## Related Publications

The Essbase documentation set includes:

- The *Start Here* booklet contains information you should read before installing Essbase, especially if you are migrating from Version 4.x.
- The *Installation Notes* contains installation instructions and tips for all system components.
- The *SQL Interface Guide* contains information on the ODBC Administrator Utility and on configuring data sources for use with Essbase.
- The *Spreadsheet Add-in User's Guide for Excel* explains how to use the Essbase spreadsheet features with Microsoft Excel for Windows. This book also includes a chapter describing how to use SQL Drill-Through profiles that an administrator sets up for other users.
- The *Spreadsheet Add-in User's Guide for 1-2-3* explains how to use the Essbase spreadsheet features with Lotus 1-2-3. This book also includes a chapter describing how to use SQL Drill-Through profiles that an administrator sets up for other users.
- The *Spreadsheet Add-in User's Guide for Excel Macintosh* explains how to use the Essbase spreadsheet features with Microsoft Excel for the Macintosh.

- The *Database Administrator's Guide* explains how to implement, manage, and maintain the Arbor Essbase OLAP server, and how to design and build Essbase applications.
- The online *Technical Reference* in your DOCS directory contains a complete listing and description of Essbase functions, macros, calculation commands, report commands, and configuration file settings.
- The online *API Reference* in your DOCS directory contains a complete listing and description of functions available through the Essbase API.



---

# Chapter 1

# A Conceptual Overview

Arbor Essbase excels at navigating large analytic sets of data rapidly and intuitively. Despite the inherent benefits of the multidimensional database for storing analytic data, some data elements that are required for analysis are better suited to remain in the relational structure.

Arbor Essbase SQL Drill-Through lets you access data from a relational data source from the Arbor Essbase Spreadsheet Add-in, and maps data from an Essbase multidimensional database to a relational database.

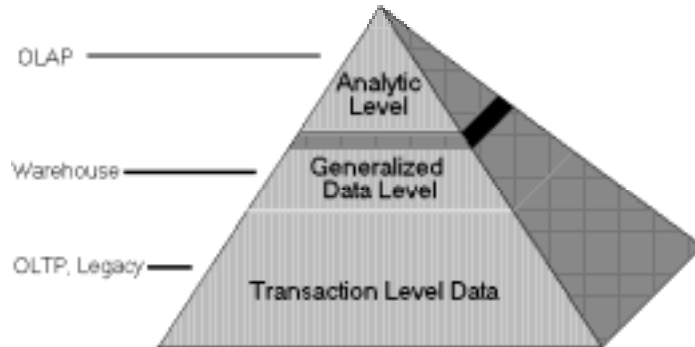
SQL Drill-Through accesses the relational data source through an ODBC (Open Database Connectivity) driver. When you install Arbor Essbase SQL Interface, as required to use Server SQL Drill-Through, the ODBC driver is automatically installed on the Arbor Essbase OLAP server. (In this document, “ODBC driver” encompasses the ODBC interface, driver manager, and driver.) If you use Client SQL Drill-Through, you must install an ODBC driver on the client machine. See Chapter 2 for more information on Client and Server SQL Drill-Through.

This chapter briefly summarizes the strengths of a multidimensional database, the characteristics of data typically stored in a multidimensional database, and how the relational and multidimensional schema can be mapped. This mapping is the key to navigating from a view of multidimensional data to relational details. The mapping is defined in SQL statements which are dynamically created as the spreadsheet user views data.

---

## Three-Tiered Data Architecture

The database architecture in many organizations uses a three-tiered approach as the optimal data storage environment, as depicted below:



The bottom layer represents what most corporations already have in place, transaction-level data that is captured in On-line Transaction Processing (OLTP) and legacy systems. This data has typically been stored in mainframe-based data sources, and more recently in relational client-server data sources.

The middle layer represents the data warehouse which is a generalized repository for transaction, summary, and miscellaneous business data.

The top layer represents the analytic data layer, and this is the On-line Analytical Processing (OLAP) component of data.

---

## The Multidimensional Model

At the heart of OLAP is a multidimensional database. Data that is stored in a multidimensional database usually represents the core analysis of a business. The core analysis that surrounds the planning and analysis process inherently has characteristics that are optimized in the Essbase architecture:

- It is hierarchical in nature. There are typically many levels of consolidation where the business must be rolled up from bottom levels to multiple points. For example, data is entered at a monthly level, then consolidated through quarterly, seasonal, and total year results.

- It is summary level. A majority of the planning and analysis that occurs in an organization is performed on summary-level data, and consolidated to total results. For example, data is planned and analyzed at a monthly level, whereas data is captured on a hourly basis in a transaction system. The transaction-level data is usually not analyzed in an OLAP environment.
- It handles complex business mathematical relationships. The math required to perform profit and loss, balance sheet, and complex ratio analysis such as product contribution are adeptly handled in OLAP.
- It contains many derived data values. The nature of hierarchical relationships between many entities implies that a great deal of derived data is created and required by OLAP. The business math that also is applied to input data values will also contribute greatly to the amount of data.
- It requires iterative data navigation. The nature of analysis among complex inter-relationships between hierarchical data points requires extensive data navigation. In order to effectively gain a whole view of how a business is performing, navigation query times are required to be in the 1-5 second range.
- It requires that multiple users update it. Just looking at numbers is a very small part of data analysis. The workgroup community of users must be able to plan their business by modifying data values and returning multiple consolidations and what-if scenarios.

---

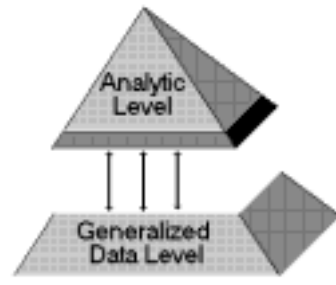
## The Relational Model

The relational model uses a database that is optimal at handling a wide range of data types, and is therefore an ideal source as a general data repository. Many organizations have built relational warehouses and more detailed transaction-level OLTP systems with the relational architecture.

---

## The Best of Both Worlds

The analysis that is designed into OLAP applications does not typically apply to transaction-level data. However, as the business is analyzed, occasionally the user needs to know the details behind a set of data. When the user requires this data, it is most efficient to query transactions that already exist in a data warehouse or other production system.



SQL Drill-Through was created to let you navigate from analytic data in multidimensional databases to transaction details in relational databases. The SQL Drill-Through facility provides intuitive data navigation into a relational database by defining mappings between the dimensional attributes of an Essbase database and the columns of relational tables. The complexity of the mapping is hidden from the end user. As with the Essbase spreadsheet interface, the end user doesn't need to learn a language and doesn't need to know how to construct Structured Query Language (SQL) queries to view relational data.

You, as the database administrator, predefine the data mapping. As the user navigates through data in their spreadsheet, the attributes of the current data item determine the SQL statement that will be sent to the relational database.

---

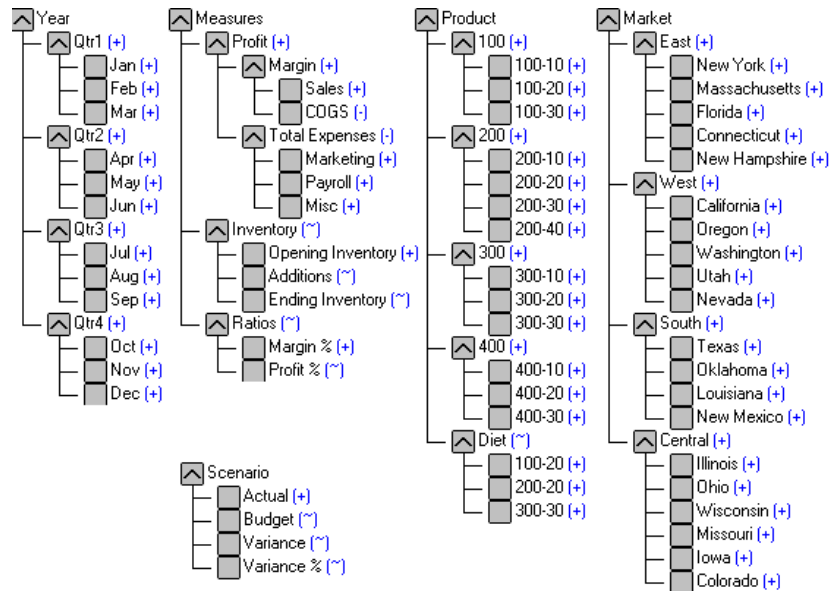
## The Conceptual Leap

To implement SQL Drill-Through, you must have a working knowledge of SQL syntax, basic relational database design, and a firm understanding of the following:

- The structure of relational tables or views that are available for the user to read. The SQL Drill-Through facility can read data from a single table, view, or tables that are joined through the SQL Drill-Through interface. The SQL Drill-Through product can connect to any relational source that is available through your client-based ODBC drivers or through Essbase SQL Interface. Refer to the *SQL Interface Guide* for more information.
- The business schema that is defined in an Essbase database outline. The Essbase outline determines the types of columns that are of potential interest to the user.
- The end-user requirements. There will not be a clear cut way of predicting how an SQL Drill-Through occurs by just understanding the source relational tables and Essbase outline. The user may have one or more requests for the scope of transaction records returned. You should understand the following:
  - The basic multidimensional navigation requirements of the end user.
  - The areas in Essbase from which the user will want to drill into more detailed data.
  - The scope of transaction records and columns that are required with each drill action.

## Multidimensional Concepts

The “multidimensional schema” is the most difficult conceptual barrier to overcome if you are familiar with relational design. The major components of an Essbase database are *dimensions* and *members*. A dimension defines a major category of data. For example, the diagram below shows five dimensions: Year, Measures, Scenario, Product, and Market.



Each dimension contains a hierarchical structure. A hierarchical structure may be flat or may contain any number of levels that define a data consolidation path. Each item within a dimension is referred to as a member. Members within a dimension are typically composed of parent-child relationships.

A single data point is comprised of the intersection of one member from each dimension. The possible maximum number of possible data cells that may be contained in a database is represented by the combination of members across all dimensions. For example, the five-dimensional model might have up to 1,065,900 potential data cells. This figure is derived by multiplying the number of members from each dimension (Year 17, Measures 19, Scenario 6, Product 22, and Market 25). Each member name in Essbase must be unique. Each Essbase data cell is a double-precision number that can be described by all of its dimensional attributes. An Essbase database with five dimensions will create each data cell with five descriptive attributes.

For example, each data value in the Sample Basic database is defined by each of its dimensional attributes:

<u>Dimensional Attributes</u>	<u>Cell Value</u>
Jan, Sales, Budget, Root Beer, Boston	120
Feb, Sales, Budget, Root Beer, Boston	132
.	
.	
Year, Sales, Budget, Product, Market	987

The way Essbase stores the data and presents the data back to the user for data navigation is much different than what is described in this basic discussion. However, for the purpose of understanding the SQL Drill-Through product, it is important to recognize that the basis for any SQL Drill-Through query is a single data cell and all of its dimensional attributes.

---

## Table Concepts

Essbase presents data to the user in a multidimensional form. This form is the most intuitive for end users to analyze their business models. Multidimensional output has the following characteristics:

- Data is oriented in either PAGE, COLUMN, or ROW dimensions.
- A single dimension can be used in only one PAGE, COLUMN, or ROW location.
- All members from a dimension are displayed in the same PAGE, COLUMN, or ROW location. This is important to note, since a dimension contains multiple levels of data (i.e., all parent-child relationships are displayed within a single PAGE, COLUMN, or ROW). In a relational table, the parent-child relationships are typically denormalized into multiple columns.

The following sheet contains a view of Essbase data:

	A	B	C	D	E	F	G	H	I
1	Root Beer								
2				Sales			Margin %		
3			Actual	Budget	Variance %	Actual	Budget	Variance %	
4	East	Jan	6760	6180	9.71	55.649968	58.09061	-4.20	
5		Feb	6920	6350	8.98	56.271676	58.89764	-4.46	
6		Mar	6921	6360	8.82	55.97457	58.49057	-4.30	
7		Qtr1	20621	18890	9.16	55.967218	58.49656	-4.32	
8	West	Jan	10436	9460	10.32	53.066727	55.49683	-4.40	
9		Feb	10564	9530	10.85	53.209012	55.50892	-4.14	
10		Mar	10674	9640	10.73	53.082256	55.18672	-3.81	
11		Qtr1	31674	28630	10.63	53.11612	55.39644	-4.12	
12	South	Jan	3976	3670	2.74	56.790744	57.62274	-1.44	
13		Feb	4082	3970	2.82	56.957374	58.19395	-3.78	
14		Mar	4055	3990	1.63	56.892725	58.64652	-2.99	
15		Qtr1	12113	11830	2.39	56.881037	58.49535	-2.76	
16	Central	Jan	10346	9970	3.77	56.150969	57.97392	-3.13	
17		Feb	10503	10150	3.48	56.38389	58.62068	-3.82	
18		Mar	10563	10210	3.46	56.385497	58.57003	-3.73	
19		Qtr1	31412	30330	3.57	56.309691	58.39103	-3.56	
20	Market	Jan	31538	29480	6.96	55.101782	57.15739	-3.60	

The PAGE dimension represented in the sheet is Product. The member from Product being viewed is Root Beer. The COLUMN dimensions represented in the sheet are Measures and Scenario. Actual, Budget, and Variance % are grouped under the Sales and Margin % measures. The ROW dimensions of the sheet are Market and Year. Note how all the hierarchical relationships are contained in a single column. That is, for the Year dimension, the quarterly data and yearly totals are in the same column, and the regions are in the same column as total market.



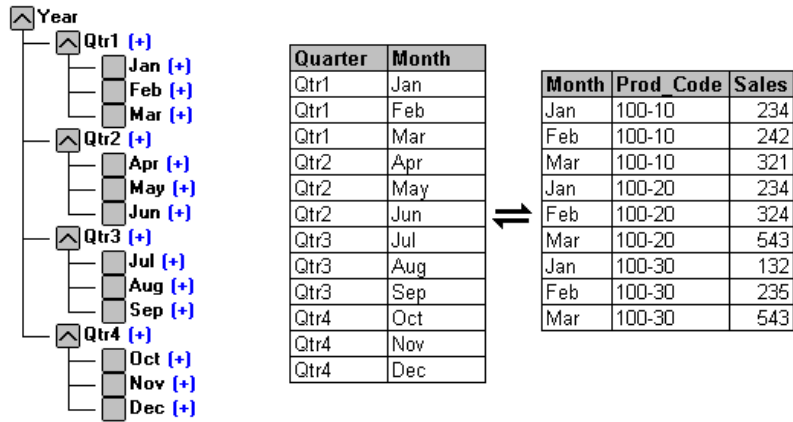
This data representation is quite different than the row and column format of a relational table. This could be viewed as:

	A	B	C	D	E	F	G	H
1	Region	Quarter	Month	Product	Scenario	Sales	Margin %	
2								
3								
4	East	Qtr1	Jan	Root Beer	Actual	6780	55.64897	
5	East	Qtr1	Feb	Root Beer	Actual	6920	56.27168	
6	East	Qtr1	Mar	Root Beer	Actual	6921	55.97457	
7	West	Qtr1	Jan	Root Beer	Actual	10436	55.96722	
8	West	Qtr1	Feb	Root Beer	Actual	10564	53.05673	
9	West	Qtr1	Mar	Root Beer	Actual	10674	53.20801	
10	South	Qtr1	Jan	Root Beer	Actual	3976	53.08226	
11	South	Qtr1	Feb	Root Beer	Actual	4082	53.11612	
12	South	Qtr1	Mar	Root Beer	Actual	4055	56.79074	
13	Central	Qtr1	Jan	Root Beer	Actual	10346	56.95737	
14	Central	Qtr1	Feb	Root Beer	Actual	10503	56.89273	
15	Central	Qtr1	Mar	Root Beer	Actual	10563	56.86104	
16	East	Qtr1	Jan	Root Beer	Variance %	9.71	-4.2	
17	East	Qtr1	Feb	Root Beer	Variance %	8.98	-4.45	
18	East	Qtr1	Mar	Root Beer	Variance %	8.82	-4.3	
19	West	Qtr1	Jan	Root Beer	Variance %	9.16	-4.4	
20	West	Qtr1	Feb	Root Beer	Variance %	10.32	-4.14	

The most differentiating characteristics of the relational and multidimensional structures are the following:

1. In a relational scheme, hierarchical parent-child relationships tend to be represented by multiple columns in a denormalized view. These are represented within a single dimension in a multidimensional model.

For example, a Year dimension typically contains all parent-child relationships such as quarter and month within the dimension. A relational database has a separate column for quarter and month.



2. A multidimensional model supports ROW and COLUMN groupings of multiple dimensions. There is no inherent grouping of related parent-child columns in a relational model. For example, the following Essbase view contains two COLUMN dimensions:

The screenshot shows an Essbase view window titled 'Book3'. The data is organized into columns for 'Product' and 'Market', and then grouped by time periods 'Qtr1' and 'Qtr2'. Each time period has sub-columns for 'Actual', 'Budget', and 'Variance'. The rows represent various financial metrics like Sales, COGS, Margin, Marketing, Payroll, Misc, Total Expenses, and Profit.

	A	B	C	D	E	F	G	H
1	Product	Market						
2								
3			Qtr1			Qtr2		
4		Actual	Budget	Variance	Actual	Budget	Variance	Actual
5	Sales	95820	89680	6140	101679	95240	6439	105215
6	COGS	42677	36140	-4737	45362	40480	-4902	47343
7	Margin	52943	51540	1403	56317	54760	1537	57872
8	Marketing	16639	11900	-3639	16716	12700	-4016	17522
9	Payroll	12168	9060	-3108	12243	9210	-3033	12168
10	Misc	233	N/A	-233	251	N/A	-251	270
11	Total Expenses	28040	20960	-7280	29210	21910	-7300	29960
12	Profit	24703	30580	-5877	27107	32670	-5763	27912

Scenarios such as Actual and Budget are grouped under each time period.

## The Mapping Point

Data is mapped by taking the dimensional attributes of a single Essbase data point that is in the spreadsheet view. The user selects a data point in the sheet that represents a multidimensional data point. The SQL Drill-Through add-in to the spreadsheet determines the dimensional attributes of a data point. For example, if the user selects cell C8 in the following sheet:

	A	B	C	D	E	F	G
1	Sales						
2	New York						
3		Actual			Variance		
4		Jan	Feb	Mar	Jan	Feb	Mar
5	100	135	187	116	(25)	(33)	(14)
6	200	162	241	88	(5)	(9)	(2)
7	300	235	220	271	(5)	(10)	1
8	400	56	232	309	5	(8)	(11)
9	Product	588	880	764	(32)	(60)	(26)
10							

The dimensional attributes of the cell are Sales, New York, Actual, Feb, and 400. These attributes become the basis of a dynamic SQL query that is hidden from the end user.

For example, a Where clause is created that maps the current member of the Year dimension to a column called MONTH in a relational table:

```
WHERE MONTH = "Feb"
```

Another example would be a Where clause that maps the current member of the Product dimension to a column called PROD\_ID in a relational table:

```
WHERE PROD_ID LIKE "400"
```



---

## Chapter 2

## Getting Started

This chapter describes how to install Arbor Essbase SQL Drill-Through. To use this product, you must have the following:

- Lotus 1-2-3 or Excel for Windows software
- Arbor Essbase Spreadsheet Add-in software

If you are using Server SQL Drill-Through, as opposed to Client SQL Drill-Through (see “Server and Client SQL Drill-Through” in this chapter), you must also have:

- Arbor Essbase SQL Interface software on the Arbor Essbase OLAP Server

SQL Drill-Through requires a Windows-based or Windows NT-based PC with the following:

---

Component	Requirement
Microprocessor	386 or higher
RAM	4 MB (6 MB recommended), 8 MB required for Windows 95
Windows Version	Windows 3.1, 3.11, Windows 95, or NT 3.5 or higher
Excel Version	Excel 5.0 or higher (5.0c is recommended)
or	Excel 95 (Version 7.0 under Windows 95)
	Excel 97 (under Windows 95)
1-2-3 Version	Release 5 or higher for Windows

---

**Note:** See the *Start Here* booklet for a list of tested and supported database drivers for use with SQL Drill-Through.

SQL Drill-Through is available in both 16-bit and 32-bit versions. You need to install the version that matches the Spreadsheet Add-in you are using.

- The Spreadsheet Add-in for Excel 95 and Excel 97 is a 32-bit version. If you install the Spreadsheet Add-in for Excel 95 or Excel 97, then be sure to install the 32-bit version of SQL Drill-Through.
- The Spreadsheet Add-in for Excel 5.0 and the Spreadsheet Add-in for 1-2-3 are 16-bit versions. If you install the 16-bit version of the Spreadsheet Add-in for Excel 5.0 or 1-2-3, you need to install the 16-bit version of SQL Drill-Through.

- If you are using Client SQL Drill-Through, make sure your ODBC driver is compatible with the Spreadsheet Add-in you are using. For example, if you are using the Spreadsheet Add-in for Excel 95 or Excel 97, then be sure to install the 32-bit version of the ODBC driver and SQL Drill-Through.

**Note:** The 16-bit and 32-bit application versions don't necessarily correspond to the operating system you are using. For example, if you are using Excel 5.0 under Windows 95, you need to install the 16-bit versions of both the Spreadsheet Add-in and SQL Drill-Through because Excel 5.0 is a 16-bit product.

---

## Server and Client SQL Drill-Through

You can access your relational data from an ODBC driver through the Essbase server, or directly from an ODBC driver on your client machine. Either way, the SQL Drill-Through user interface is the same.

**Server SQL Drill-Through** sends user requests to remote databases through the Essbase server. You need only install SQL Interface on the server and configure the resulting ODBC driver.

**Client SQL Drill-Through** sends user requests directly to remote databases. It is faster than Server SQL Drill-Through, but you need to properly install, configure, and maintain an ODBC driver on every user's client machine. You must purchase the ODBC driver separately from your choice of vendor.

The `SQLDrillServer` keyword in the `SQLDRILL.INI` file determines how your source data is accessed. Server SQL Drill-Through is the default. Only a database administrator should modify the `SQLDRILL.INI` file. See Appendix A for more information about the `SQLDRILL.INI` file.

---

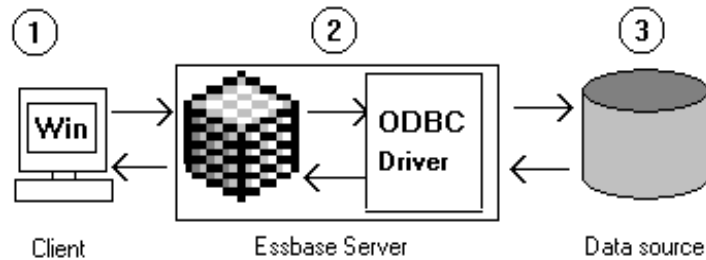
## SQL Drill-Through Architecture

As discussed in the previous section, you can use Server SQL Drill-Through or Client SQL Drill-Through to access your data.

---

### Server SQL Drill-Through

Server SQL Drill-Through (the default) routes requests as follows:

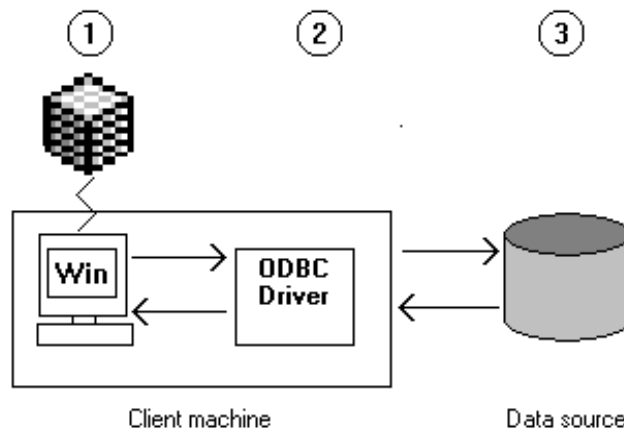


1. A user requests data from the client machine using SQL Drill-Through. The SQL Drill-Through product is installed on a Windows 3.1, Windows 95, or Windows NT client machine that already has the Spreadsheet Add-in for 1-2-3 or Excel for Windows. The `SQLDRILL.INI` file is installed in the Windows directory and lists the Essbase-to-relational database mappings.
2. SQL Drill-Through sends the request from the Windows client and through the Essbase server to the data source. The SQL Interface software, which includes the ODBC drivers, must be properly installed on the server to connect to a data source.
3. When the SQL query (request) is sent, the user connects to the data source, supplying a valid user name and password if necessary. The security to the data source is controlled by the data source. The relational data sources must be accessible to the server.

---

## Client SQL Drill-Through

Client SQL Drill-Through routes requests as follows:



1. A user requests data from the client machine using SQL Drill-Through. The user must be connected to an Essbase server to use SQL Drill-Through. The SQL Drill-Through product is installed on a Windows 3.1, Windows 95, or Windows NT client machine that already has the Spreadsheet Add-in for 1-2-3 or Excel for Windows. The `SQLDRILL.INI` file is installed in the Windows directory and lists the Essbase-to-relational database mappings.
2. SQL Drill-Through sends the request through the ODBC driver to the data source. The ODBC driver must be installed on the client machine.
3. When the SQL query (request) is sent, the user connects to the data source, supplying a valid user name and password if necessary. The security to the data source is controlled by the data source. The relational data sources must be accessible to the Spreadsheet Add-in user.



---

## Installing SQL Drill-Through

These instructions apply to both Server and Client SQL Drill-Through unless otherwise noted.

1. Register or enable the Essbase server software.

The Essbase server controls your ability to use the SQL Drill-Through product. You need to enable the SQL Drill-Through option using the software license number assigned to your Essbase server.

- If you are reinstalling the server software at the same time as you are installing this version of SQL Drill-Through, then SQL Drill-Through will be enabled automatically, based on your license number.
- If you aren't reinstalling the server software, then you simply need to enable SQL Drill-Through. Run the REGISTER.EXE program on the server and specify your new license number. See the *Installation Notes* for more information.

In addition to registering SQL Drill-Through on the server, SQL Drill-Through users must install both the Spreadsheet Add-in and SQL Drill-Through software on their local client machines. Users must install both into the same directory.

2. Run the installation program from the Essbase CD-ROM. The installation program displays a menu that lets you choose server and client options, including SQL Drill-Through.

Insert the Arbor Essbase CD-ROM. From the Windows Program Manager, choose File | Run, and then type

```
d:setup
```

and press Enter. (If you use drive E, type e:setup and press Enter.)

3. Follow the directions on your screen. When prompted for the directory name, specify the directory name that contains the Spreadsheet Add-in. The default name is C:\ESSBASE.

**Note:** To install SQL Drill-Through to a network drive, see "Installing SQL Drill-Through on a Network."

4. Open either 1-2-3 or Excel for Windows. A new menu command, SQL Drill-Through, appears when you view the Essbase menu in your spreadsheet environment.
5. Decide whether you want Server or Client SQL Drill-Through.

If you use Server SQL Drill-Through, Essbase sends all SQL statements to the ODBC driver via the Essbase server. If you use Client SQL Drill-Through, Essbase sends all SQL statements directly to the ODBC driver. The `SQLDrillServer` keyword in the `SQLDRILL.INI` file determines how your source data is accessed. Server Drill-Through is the default. For more information, see Appendix A.

6. If you are using Client SQL Drill-Through:
  - a. Install your ODBC driver on your client machine. Essbase does not provide the ODBC driver.
  - b. Make sure your ODBC driver is compatible with the spreadsheet environment you are using. For example, if you are using Spreadsheet Add-in for Excel 95 or Excel 97, then be sure to install the 32-bit versions of your ODBC driver and SQL Drill-Through.
  - c. Before you use SQL Drill-Through, test the connection between your ODBC and the remote databases it should access.

If you are using Server SQL Drill-Through:

Before you use SQL Drill-Through, install the Essbase SQL Interface product on the server and test the connection. See the *Installation Notes* for more information. The SQL Interface installation includes sample dBASE files that you can use with the examples in this book and the *Spreadsheet Add-in User's Guide*.

7. After installation, create SQL Drill-Through mapping profiles to fit users' mapping requirements for the databases they are using. The `SQLDRILL.INI` file contains the mapping profile. You must put an appropriate `SQLDRILL.INI` file on the client machine of every SQL Drill-Through user, whether you are using Server or Client SQL Drill-Through. You may need to create several versions of the file, depending on individual user needs. See Chapter 3 for information.

**Note:**

If you are using Server SQL Drill-Through, it is recommended that you use an INTERSOLV ODBC driver provided with the SQL Interface software. If you replace the default ODBC driver or any of its components or associated software, connectivity might be unstable and is unsupported.

---

## Installing SQL Drill-Through on a Network

You cannot install SQL Drill-Through *to* a network drive using the network installation feature of the CD-ROM installation program. If you select the **Network** option in the **Client Setup and Destination Directory** dialog box, SQL Drill-Through does not install correctly.

Instead, do one of the following to enable your client users to install SQL Drill-Through *from* the network drive:

- If you want your client users to set up their client PC's operating environment to run SQL Drill-Through on the network drive, you need to install SQL Drill-Through on the network drive. Use the **Local** option in the **Client Setup and Destination Directory** dialog box, and then choose a network directory. For example, choose **Local**, then choose the directory `N:\ESSBASE`.
- If you want your client users to install SQL Drill-Through on their client PC's hard drives, use your operating system to copy the contents of the SQL Drill-Through directory from the CD-ROM to a network drive. You can copy the contents to any network directory. If you have previously installed the Spreadsheet Add-in or the Application Manager to a network drive, you can copy the installation disk set for SQL Drill-Through to an equivalent Essbase directory on the same network drive.

The installation disk set for SQL Drill-Through is in one directory called `DISK1`. There is a `DISK1` for the 32-bit SQL Drill-Through, and a `DISK1` for the 16-bit SQL Drill-Through.

To copy the `DISK1` directory and its contents to a network drive, using your operating system, do the following:

1. Create a directory for SQL Drill-Through that is at the same level as the installation disk set directories created by the **Network** installation program for the Application Manager and Spreadsheet Add-in.

For example, if you install both the Application Manager, and the Spreadsheet Add-in for Excel 95, to the `ESSBASE` directory on the network drive `N`, the **Network** installation program creates the following directories:

```
N:\ESSBASE
    APPMAN
    BIN
    CLIENT
    ESSEXC32
```

The installation disk set for the Application Manager is in the `APPMAN` directory, and the installation disk set for the 32-bit Spreadsheet Add-in for Excel 95 (or Excel 97) is in the `ESSEXC32` directory.

Create an equivalent directory for the 32-bit SQL Drill-Through installation disk, and call the directory `ESSSQL32`.

```
N:\ESSBASE
    APPMAN
    BIN
    CLIENT
    ESSEXC32
    ESSSQL32
```

2. Using your operating system, copy the contents of the SQL Drill-Through product directory to this directory.

For example, using DOS, type the following command:

```
XCOPY D:\WIN32\SQLDRILL\*.* N:\ESSBASE\ESSSQL32\*.* /S
```

This copies the installation disk set for the 32-bit SQL Drill-Through, a directory called `DISK1`, and its contents, from the CD-ROM on drive `D` to the Essbase directory `ESSSQL32` on drive `N`.

## Verifying an SQL Connection for Server Drill-Through

The SQL Interface product comes with sample dBASE files that you can use to verify a proper connection. Once you install the SQL Drill-Through product, use the following steps to verify that all components are installed correctly.

1. Configure dBASE as a valid data source through the ODBC Administrator Utility on your Essbase server machine. The *SQL Interface Guide* provides information on how to define an ODBC data source.

**Note:** The sample dBASE files used in this book are installed in the \ESSBASE\APP\SAMPLE directory of your server machine.

2. Open either 1-2-3 or Excel for Windows. You use the Spreadsheet Add-in to access the SQL Drill-Through product.
3. If you are using 1-2-3, choose File | Open and select the sheet SQLDRILL.WK4. If you are using Excel for Windows, open the sheet SQLDRILL.XLS.

These sheets are supplied as part of the Spreadsheet Add-in installation and are located in the \ESSBASE\CLIENT\SAMPLE directory.

	A	B	C	D	E	F	G	
1	Sales							
2	New York							
3			Actual				Variance	
4		Jan	Feb	Mar	Jan	Feb	Mar	
5	100	135	187	116	(25)	(33)	(14)	
6	200	162	241	68	(6)	(9)	(2)	
7	300	235	220	271	(5)	(10)	1	
8	400	56	232	309	6	(8)	(11)	
9	Product	588	880	764	(32)	(60)	(26)	
10								

**Note:** SQLDRILL.XLS is an Excel 4 file.

4. Choose Essbase | Connect and connect to the Sample Basic database.

This is supplied with the Essbase server software installation.



---

## Verifying an SQL Connection for Client Drill-Through

Before you proceed with Client SQL Drill-Through, make sure you can access remote data with your ODBC driver *without* using SQL Drill-Through. Use your database client's connectivity tools; for example, if you are using Sybase, Oracle, or Informix ODBC drivers or Microsoft Access, use the Microsoft Query tool.

---

### The First Thing to Verify

You must run 16-bit applications (for example, Excel 5.0) with 16-bit ODBC drivers, and 32-bit applications (for example, Excel 95 or Excel 97) with 32-bit ODBC drivers. Also, use the correct version of the Windows ODBC Administrator for the application and driver you are using; 16-bit and 32-bit versions of the ODBC Administrator might exist on your client machine.

To verify an ODBC driver version:

1. Open the Windows Control Panel.
2. Double-click the ODBC Administrator.
3. Choose **Drivers**.
4. Select a driver (single-click it) and click **About**.

---

### Examples

These examples assume you have properly installed an ODBC driver on your client and that you have access to the appropriate data sources.

---

#### *Testing a Connection with Microsoft Query*

Microsoft Query is available with Microsoft Office Professional; use the customized installation, because the Microsoft Query does not get installed by default.

This example is applicable for Windows 3.11 and Windows 95.

**Note:** Microsoft Query might not expose 16-bit and 32-bit incompatibilities.

1. Open Microsoft Query and choose File | New Query.

The **Select Data Source** dialog box appears.

2. If the desired data source appears in the dialog box list, select the data source and then click **Use**.

If the desired data source does not appear, click **Other** and choose from the resulting list of ODBC data sources. (You might need to click **New** to add a new data source; see Microsoft Query documentation for details.)

3. Choose a table from the **Add Tables** dialog box.
4. Click a field in the resulting **Table Pane** list.

Microsoft Query runs the query and retrieves the data.

If you cannot successfully connect to your ODBC data source using Microsoft Query, see your database administrator. For more information about using Microsoft Query, see your Microsoft Query documentation.

---

### *Testing a Connection with an Oracle ODBC Driver*

This example is applicable for an Oracle 7.x ODBC driver. It assumes you have installed SQL\*Net for Windows as required for this driver, and that you already have access to an Oracle 7 ODBC data source.

**Note:** Any vendor tools, such as the Oracle tools described here, might change in name or function from version to version. If you cannot locate the tool or make it work, contact your ODBC vendor.

Oracle ODBC drivers provide a program, `ODBCSTST.EXE`, for testing your ability to connect to an Oracle 7 server. *Remember to test 32-bit ODBC drivers with 32-bit testing tools, and 16-bit ODBC drivers with 16-bit tools.*

1. Run `ODBCSTST.EXE`. You can double-click its icon, or choose File | Run (Windows 3.x) or Start | Run (Windows 95).

The file is normally installed into the ODBC group when you install the Oracle driver; if it is not there, look in the `\WINDOWS\SYSTEM` directory.

2. Choose **Connect** from the ODBCSTST toolbar.



3. Select a data source name from the **Data Sources** list box and click OK.
4. Provide a valid user name and password when prompted, and click OK.
5. If you didn't see any error messages, you have probably connected successfully to your Oracle 7 server.

If you connect successfully, Oracle displays two windows. Execute an SQL SELECT statement from the top window; the bottom window should display the returned data.

You also might want to use other native Oracle tools such as SQL\*Plus, TNSPING (SQL\*Net V2), or NETTEST (SQL\*Net V1). See your Oracle ODBC documentation for instructions.

---

### *Testing a Connection with a Sybase ODBC Driver*

Sybase users need to write their own Open Client programs to test connectivity. Sample Open Client programs, such as ISQL, are provided with some versions of Sybase ODBC Open Client software. Refer to your Sybase documentation for details and instructions.

---

## The SQLDRILL.INI File

The SQL Drill-Through installation places a file named `SQLDRILL.INI` in the `\WINDOWS` directory. The `.INI` file stores the Essbase-to-relational mappings. These mappings are called *profiles*. It contains two sample SQL Drill-Through profiles to use with the Sample Basic database. In practice, once you set up a mapping profile for a real application, you will take an updated `.INI` file and copy it over the `.INI` file that is installed on each user's desktop.

**Note:** Only a database administrator should modify the `SQLDRILL.INI` file.

The `SQLDRILL.INI` file contains a setting that enables a `Supervisor` privilege. This privilege provides the database administrator with the ability to create, edit, and delete mapping profiles. The privilege can also be used by advanced users to edit or create their own profiles. Mapping profiles are defined through a series of dialog boxes that update the `.INI` file. The default installation has the `Supervisor` option turned on.

The `SQLDRILL.INI` file also contains a setting that determines whether you run Server or Client SQL Drill-Through. If you use Server SQL Drill-Through, Essbase sends all SQL statements to the ODBC driver via the Essbase server. If you use Client SQL Drill-Through, Essbase sends all SQL statements directly to the ODBC driver. The `SQLDrillServer` keyword in the `SQLDRILL.INI` file determines how your source data is accessed. Server SQL Drill-Through is the default.

For more information about the `.INI` file, refer to Appendix A.



2. A database administrator or power user that needs to define a new mapping or modify an existing mapping should have an `.INI` file with the `Supervisor` privilege enabled.
3. The SQL Drill-Through add-in dialog boxes modify the current `SQLDRILL.INI` file, if the `Supervisor` flag is enabled. Before distributing your `SQLDRILL.INI` to each user, edit the file and disable the `Supervisor` setting, if necessary. See Appendix A for more information.

---

## Designing a Profile

This section describes how to design an SQL Drill-Through mapping profile. Use the following sequence of steps to provide end users with SQL Drill-Through capability:

1. Review the Essbase database schema and interview the end user(s) for drill-through requirements. This involves reviewing the Essbase database outline and reviewing how the end user navigates through data in the spreadsheet.  
  
A thorough understanding of the dimensions and levels within each Essbase dimension is crucial to defining a mapping relationship.
2. Set up an SQL Drill-Through profile which contains the relevant mapping information. All mapping information is defined through Spreadsheet Add-in dialog boxes. All actions recorded in the dialog boxes are captured to the `SQLDRILL.INI` file.
3. Copy the `.INI` file to the machine of all users that require the same mapping relationships. Before copying the `.INI` file to each user's desktop, set the `Supervisor` setting to 0. See Appendix A for more information.

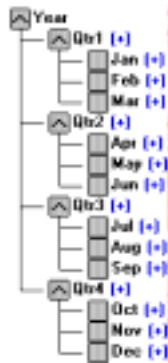
---

## Understanding End-User Requirements

This section presents three different end-user requirements that are based on the same Essbase database, called Sample Basic. The Sample Basic database is shipped with the Arbor Essbase OLAP Server and is used as an example in all Essbase documentation.

When defining an SQL Drill-Through mapping, the database administrator must be familiar with the Essbase outline. The Sample Basic database outline has five dimensions. Each is explained below.

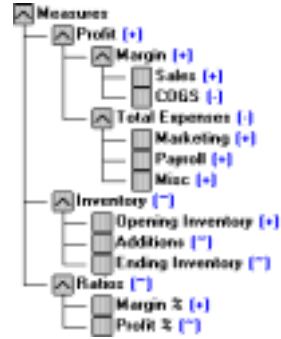
The Year dimension contains a hierarchy of time-related members:



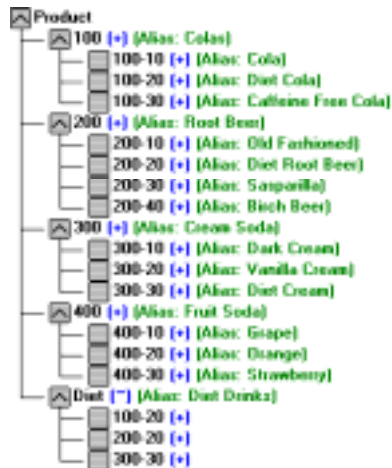
In Essbase terminology, the different levels within a dimension are referred to as *generations*. The very top of the dimension, in this case Year, is referred to as generation 1. Quarterly members are at generation 2; monthly members are at generation 3.

Each level within the dimension represents a different parent-child relationship. For example, months are children of quarters and quarters are children of total year. As you will see below, many applications use a Scenario dimension to define slices of specific years (such as 1996 or 1997) or data scenarios (such as 1997 Budget).

The Measures dimension represents the various financial measures that are relevant for analysis. Financial measures do not usually have a natural hierarchy where data is consolidated from the bottom up. However, Essbase lets the application designer create hierarchical relationships that are used for data navigation.

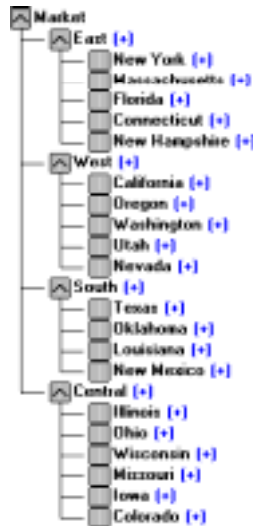


The Product dimension contains members that define product families and product groups.



Note that each product code is a member name, and that each member has an alias name (an alternate description for reporting purposes). Also of particular interest are products in the Diet category, where children under Diet are also children of their main product families. This is an example of a shared member hierarchy where members roll up to multiple parents.

The Market dimension contains a straightforward roll up.



States roll up to their respective parent region, and regions roll up to Market.

The Scenario dimension is defined as a flat hierarchy.



The Scenario dimension is typically used in most Essbase applications to define data slices for each year or data type such as 1996 Actual, 1997 Actual, 1998 Budget, 1997 Forecast, and Plan.

Understanding the basic dimensional structure and parent-child relationships within the dimensional schema are important to translating the user's needs to access data from a relational source. When determining the end-user requirements for SQL Drill-Through, you need to understand what the user expects to see for each Essbase dimension and for each level within those dimensions.

The examples in this book use four tables as the source for the transactional data required by the user.

**AREA** contains the following columns:

PROD_CODE	Contains codes down to the product ID level. These codes are in the form of xxx-yy-zz, where xxx is the product family, yy is the product group, and zz is the package/size code.
PACKAGE	Contains a package/size description.
DESCRIPTION	Contains a product group description.
STATE	Contains the name of the state where the transaction occurred.
AREA	Contains the name of the metropolitan area within the state. These can be thought of as children of the state.
MONTH	Contains the month of the transaction.
UNIT_SALES	Contains data for unit sales.
RETURNS	Contains data for returns.

**WEEKLY** contains the following columns:

PROD_FAM	Contains product codes at the product family level.
STATE	Contains the name of the state where the transaction occurred.
W_ENDING	Contains the week-ending date of the transaction.
UNIT_SALES	Contains data for unit sales.

**CALENDAR** contains the following columns:

MONTH	Contains month names.
W_ENDING	Contains week-ending dates that correspond to the fiscal month.

**TERRITOR** contains the following columns:

REGION	Contains geographic region names.
STATE	Contains state names that correspond to a given region.



---

## Sample Requirements

This section describes several user requirements. For each requirement, design issues are reviewed, and the appropriate dialog boxes are used to define mapping information.

The sample application uses a spreadsheet file that is supplied with the Spreadsheet Add-in installation. For Excel, the file is called `SQLDRILL.XLS`. For 1-2-3, the file is called `SQLDRILL.WK4`. These are located in the `\ESSBASE\CLIENT\SAMPLE` directory on the drive in which you installed the Essbase Spreadsheet Add-in.

---

### Requirement 1

The Sample Basic application contains data that is used for product planning. When performing variance analysis between actual and budget data, users occasionally need to view more detailed information about product IDs and market areas. Neither of these details are stored in the Essbase database. On a dimension-by-dimension basis, users require the following:

- **Product Dimension.** When users view information about product families or product groups, transactions by Product ID must be returned. The retrieved records must match the code of the product currently being viewed in the Essbase model.
- **Market Dimension.** Users want to view transactional data only when they are looking at the lowest level in the Market dimension. In this case, users want to view area data only for the state they are currently viewing from Essbase.
- **Year Dimension.** Users want to view transactional data only when they are looking at quarterly or monthly data. In this case, users want to view monthly data that relates to the period they are currently viewing from Essbase.
- **Scenario Dimension.** The Scenario data in Essbase is irrelevant because users always want to view actual figures from the relational database.
- **Measures Dimension.** The Measures data in Essbase is irrelevant because users always want to view unit sales and returns from the relational database.

To further define the drill-through requirement, users show the following spreadsheets and describe the desired behavior:

	A	B	C	D	E	F	G
1	Sales						
2	New York						
3			Actual			Variance	
4		Jan	Feb	Mar	Jan	Feb	Mar
5	100	135	187	116	(25)	(33)	(14)
6	200	162	241	68	(8)	(9)	(2)
7	300	235	220	271	(6)	(10)	1
8	400	50	232	309	6	(8)	(11)
9	Product	588	880	764	(32)	(50)	(26)

By selecting cell C8 (which, in this example, has data cell attributes of Sales, New York, Actual, Feb, 400), a user wants to see all transactions that are within the New York area, within product family 400, and for the month of February.

	A	B	C	D	E	F	G
1	Sales						
2	New York						
3			Actual			Variance	
4		Jan	Feb	Mar	Jan	Feb	Mar
5	400-10	234	232	234	(16)	(18)	(16)
6	400-20	219	243	213	(11)	(17)	(17)
7	400-30	134	189	198	(6)	(11)	(12)
8	400	587	664	645	(33)	(45)	(45)
9	Product	2479	2625	2601	179	175	161

If a user selects cell C6 (which has data cell attributes of Sales, New York, Actual, Feb, 400-20), the user wants to see all transactions that are within the New York area, within product group 400-20, and for the month of February.

An SQL Drill-Through profile must be defined that will query the source table or view. In this example, the AREA table already has all the elements required to return the appropriate data.

To define an SQL Drill-Through profile, you, as the database administrator, need to take the following steps:

1. Start an Excel or 1-2-3 session, and connect to the appropriate Essbase database.

This book uses sample data based on the Sample Basic database.

2. Display the data that is representative of how your users will be viewing data. From this display, you will define all the dimensional attributes of data items to be used as the basis of a drill action.

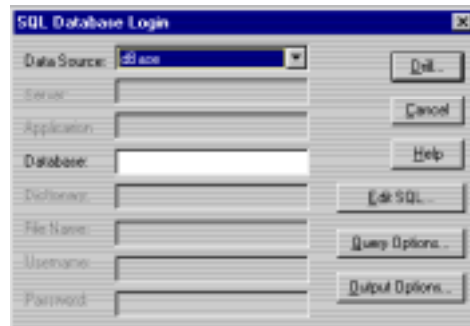
For this example, open `SQLDRILL.XLS` or `SQLDRILL.WK4` in the `\ESSBASE\CLIENT\SAMPLE` directory on the drive in which you installed the Spreadsheet Add-in.

3. Select a data cell that is representative of how your users will use SQL Drill-Through.

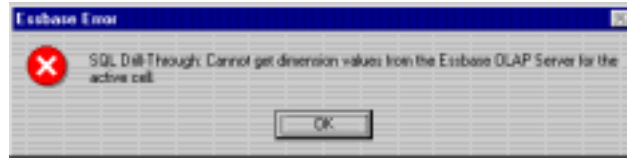
For this example, select cell C8 (the data cell having attributes of Sales, New York, Actual, Feb, 400) from the sample spreadsheet.

4. Choose Essbase | SQL Drill-Through.

A short pause occurs while the Essbase server determines the dimensional attributes of the cell you selected in the spreadsheet. The Essbase client-server icon appears while the server updates the client session with the relevant information. The **SQL Database Login** dialog box appears:



If the current cell doesn't represent a data point from the multidimensional model, the following error message appears:



If this occurs, click OK. Then select a cell that contains data from an Essbase retrieval, and choose Essbase | SQL Drill-Through again.

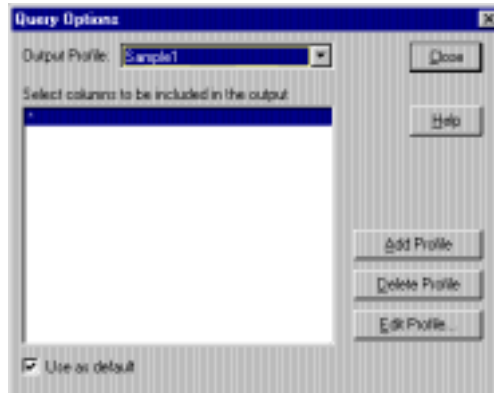
The **SQL Database Login** dialog box contains any login information saved from a prior session. If you open the dialog box for the first time during your current spreadsheet session, then you might need to enter a password to access the RDBMS. For more information on the **SQL Database Login** dialog box, click Help.

**Note:** For Server SQL Drill-Through, the available data sources are defined by SQL Interface, and each mapping profile can connect to only one type of data source type. For more information, see the *SQL Interface Guide*.

5. Click **Query Options**.

Essbase attempts to connect to the data source. If successful, the entries in the **SQL Database Login** dialog box, with the exception of the password, are saved to the `SQLDRILL.INI` file.

The **Query Options** dialog box appears:



The **Query Options** dialog box lets you create, edit, or delete a profile. If you select an existing profile, the list box specifies which data columns from the data source are available to be retrieved. The **Use as default** check box specifies that the currently selected columns in the list box should be the default columns for any drill request. For more information on the **Query Options** dialog box, click **Help**.

**Note:** Output profile names are *not* case-sensitive. Do not give multiple profiles the same name with different cases.

Two sample profiles come with the SQL Drill-Through product in the `SQLDRILL.INI` file. These are used in this book and in the *Spreadsheet Add-in User's Guide*. Select **Sample1** from the **Output Profile** list box.

**Note:** The **Add Profile**, **Delete Profile**, and **Edit Profile** buttons are available only if you are defined as a **Supervisor** in the `SQLDRILL.INI` file. See Appendix A for more information.

6. To create a new profile, click **Add Profile**.

The **Add New Profile** dialog box appears:



7. Check **Use current profile values as defaults**. This copies the definition from the selected profile in the **Query Options** dialog box to the new profile.
8. Enter `SKU MSA Details` as the name, then click **OK**.

**Note:** A profile name can be up to 42 alphanumeric characters.

- From the **Query Options** dialog box, select **SKU MSA Details** in the **Output Profile** list box, then click **Edit Profile**.

The **Profile Editor** dialog box appears:



Since the defaults from `Sample1` were copied to `SKU MSA Details`, the profile information already appears in the dialog box.

The **Profile Editor** dialog box lets you define mapping information on three pages of the dialog box:

- SQL Generator
- Defined Columns
- User Exit

**Note:** For more information on the pages of the **Profile Editor** dialog box, click **Help**.

The **SQL Generator** page defines the database mapping for each dimension. To map a dimension from the currently connected database, first choose it from the **SQL Rules for** list box.

For each dimension, you can specify a minimum generation number that the user must be viewing before an SQL statement is formed. Check **User must attain at least generation**, and then enter the generation number. If this option is checked but the user hasn't attained the minimum generation number, you have two options:

- **Don't Generate Where Clause** prevents an SQL Where clause from being generated for the given dimension, unless the user attains the specified generation number. The Where clause defines how to select records for the currently selected dimension. Note, however, that an SQL statement can still be executed based on other dimensions in the database.
- **Don't allow SQL Drill-Through** doesn't send the SQL query to the data source, unless the user attains the specified generation number.

If you clear the check box, an SQL statement will be executed, as defined by the SQL generation rules, regardless of the current level of data that the user is viewing within the given dimension.

**Note:** For more information about SQL generation rules and syntax, click Help.

10. Define the mapping rules for each dimension.

For the Market dimension, the SQL Drill-Through should occur only if the deepest generation number of the Market dimension is attained. In the Sample Basic database, the deepest generation number is 3, which contains state members.

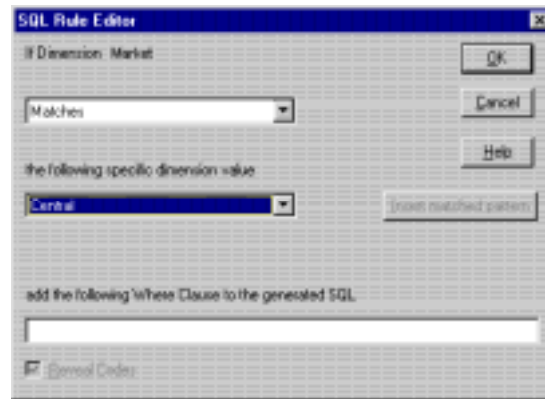
To define the mapping rule for Market:

- a. Choose **Market** from the **SQL Rules for** list box.
- b. Check **User must attain at least generation**, and enter a value of 3.
- c. Choose **Don't allow SQL Drill-Through**. If the user attempts a drill action on a member of Market that is higher than generation 3, then no SQL Drill-Through action can occur.



- d. Define the SQL generation rules. An SQL generation rule defines a Where clause to be created for the current dimension. To define a rule, click **New**. To edit an existing rule, select the rule and click **Edit**.

The **SQL Rule Editor** dialog appears:



**Note:**

The rules described in this step were defined in `Sample1`, so you don't need to recreate them.

In this dialog box, you select one of three options that, when met, trigger the Where clause to be created.

- The **Matches** option matches the dimension data to a specific value.
- The **Matches Pattern** option matches the dimension data to a pattern match string. The string can contain any specific character or the following wildcard characters: \* matches any string; ? matches any character.
- The **Does not Match Pattern** option rules out dimension data that matches a pattern match string.

If you choose **Insert matched pattern**, a numbered string (`\1`) is inserted into the Where clause text box. The string is replaced with the member name from the current dimension when the query runs. For example, the `AREA` table has a column called `STATE`. This is the column used as the basis for the Where clause related to the Market dimension. The rule copied from the `Sample1` profile defines a Where clause of `STATE = '\1'`. This causes the currently selected member of the Market dimension to be substituted into the Where string. If the active data cell is represented by New York, then the Where clause becomes `Where STATE = 'New York'`.

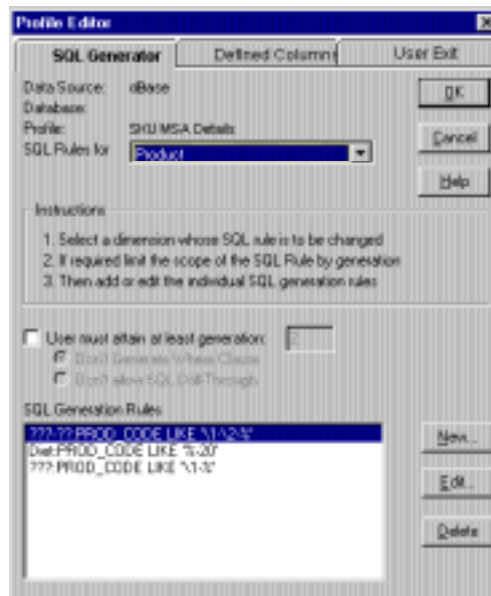
When the Where clause is triggered, the **add the following Where clause to the generated SQL** text box specifies the column to which the Where clause applies. Type the column name.

If you check **Reveal Codes**, the current dimensional value of the active data cell is placed into the substitution string.

For the Product dimension, the SQL Drill-Through should occur only if the product-family generation or deeper is attained. In the Sample Basic database, the product-family generation starts at generation 2.

To define an SQL rule for Product:

- Choose **Product** from the **SQL Rules for** list box.
- Check **User must attain at least generation**, and enter a value of 2.
- Choose **Don't allow SQL Drill-Through**.
- Define the SQL generation rules. The `PROD_CODE` column is used for the mapping. This column contains sub-strings that match the Product member names. For example, a `PROD_CODE` value of 400-10-12 is composed of a family code of 400, a group code of 10, and a size code of 12.



- The first rule uses specific wildcard characters in the xxx-yy group code. If the active data cell has a Product attribute in this form, such as 400-10, then a Where clause of `PROD_CODE LIKE 400-10-%` is created. Records that have a product ID with a starting code of 400-10- will satisfy this rule in the query.
- The second rule checks for a member name of Diet. Only products with a group code of 20 are part of the Diet family. Therefore, a Where clause of `PROD_CODE LIKE %-20-%` is used.
- The third rule is used when the active data cell has a family code attribute, such as 400. If the active data cell has a product-family attribute in this form, then a Where clause of `PROD_CODE LIKE 400-%` is created. Records that have a product ID with a starting code of 400- will satisfy this rule in the query.

**Note:**

When multiple rules are defined for a single dimension, the rules are validated from the top of the list to the bottom. The first rule to be successfully matched is used to generate the Where clause.

For the Year dimension, the SQL Drill-Through should occur only if the generation with the quarterly data is attained. In the Sample Basic database, the quarterly generation is number 2.

To define an SQL rule for Year:

- a. Choose **Year** from the **SQL Rules for** list box.
- b. Check **User must attain at least generation**, and enter a value of 2.
- c. Choose **Don't allow SQL Drill-Through**.

- d. Define the SQL generation rules. The MONTH column is used for the mapping.



- The first four rules are defined for a data cell with a quarterly attribute. If the active data cell is for Qtr1, then a Where clause of `MONTH = 'Jan' OR Month = 'Feb' OR MONTH = 'Apr'` is created.
- The final rule uses a 3-digit wildcard pattern to match a member name from the Year dimension. In this case, the member name for the active data cell is substituted into the string in place of \1. For example, if the active data cell has an attribute of Jan, then the Where clause becomes: `MONTH = 'Jan'`

**Note:** Based on user requirements, the Measures and Scenario dimensions do not require SQL Drill-Through mapping rules.

- Once the SQL rules are defined, switch to the **Defined Columns** page of the **Profile Editor** dialog box. This lets you define the columns available to the user from one or more relational tables or views.



- The **Tables in Database** list box shows a list of all tables defined for the data source. In the case of dBASE, all .DBF files located in the directory specified by the ODBC Administrator Utility are listed. To add a table to the profile you are editing or creating, select the table name, and click **Copy->**.
- The **[Define Table]** option in the **Tables in Database** list box lets you define a table from the data source. See the “Requirement 2” section for more information.
- The **Table Link** button lets you join multiple tables. This button is unavailable if multiple tables are not present. See the “Requirement 2” section for more information.
- The **Tables in profile** list box shows all tables defined in the profile you are editing or creating. You can remove a table from the profile by selecting the table name and clicking **Delete**.

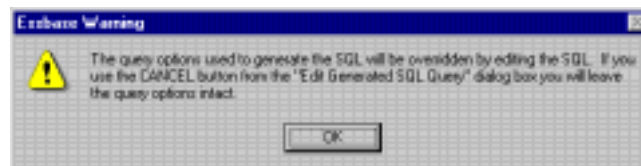
**Note:** You cannot delete a table that has a defined table link or that has a column in the **Defined Column List**.

Double-click a table name in the **Tables in profile** list. The columns in the table appear in the **Columns in selected Table** list box. Specify which columns can be viewed by the user by selecting a column and adding it to the **Defined Column List** by clicking **Copy->**. You can remove a column from the list by selecting the column name and clicking **Delete**. Check **Allow Star** if you want to define a `SELECT *` statement. This allows the user to view all columns from all tables in the **Tables in profile** list.

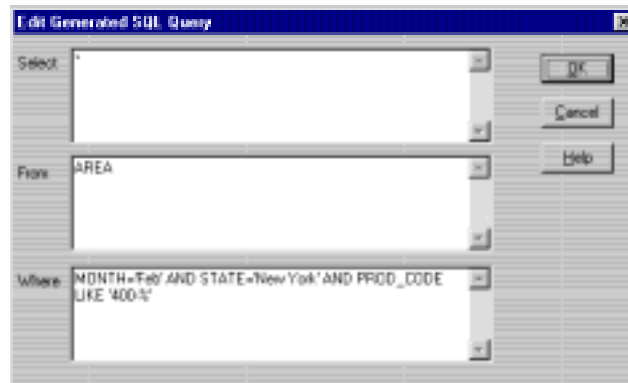
**Note:** For more information on defining columns, click Help.

12. Click OK when all table and column definitions are defined.
13. Close the **Query Options** dialog box to return to the **SQL Database Login** dialog box.
14. Click **Edit SQL** to view the SQL statement created by the profile definition.

The following message appears:



15. Click OK. The **Edit Generated SQL Query** dialog box appears:



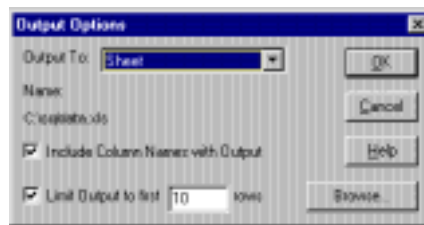
- The **Select** clause is created from the list of columns in the **Query Options** dialog box.
- The **From** clause is created from the list of tables on the **Defined Columns** page of the **Profile Editor** dialog box.
- The **Where** clause is generated dynamically from the attributes of the active data cell, and the mapping rules defined on the **SQL Generator** page of the **Profile Editor** dialog box.

**Note:**

The **Edit Generated SQL Query** dialog box is an excellent tool both for verifying that the resultant SQL statement selects the appropriate table information and for viewing the results of the SQL query several different ways in the spreadsheet.

16. Click OK to return to the **SQL Database Login** dialog box and choose **Output Options**.

The **Output Options** dialog box appears:

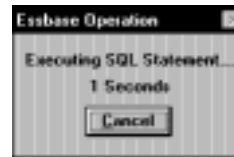


17. Select a destination for the result of the SQL query using the **Output To** list box:
  - The **File** destination stores the records in an ASCII text file. Choose **Browse** to specify an existing or new file name.
  - The **Printer** destination sends the data to a selected printer.
  - The **Sheet** destination stores the data in an .XLS or .WK? file that corresponds to the version of Excel or 1-2-3 you are using. Choose **Browse** to specify an existing or new file name.
18. To limit the number of records that are returned by the query, check **Limit Output**, and enter the maximum number of rows.
19. Choose **Sheet** and click OK.

Once you have verified that the SQL statement is correct for the active data cell, and you have set the output options, you are ready to test SQL Drill-Through.

20. To begin an SQL Drill-Through action, click **Drill**.

The following dialog box appears:



The Essbase server passes the SQL statement to the source database. The query operation is asynchronous, so you can cancel it at any time.

The data appears as follows:

	A	B	C	D	E	F	G	H
1	PROD_CD	PACKAGE	DESCRIPTIO	STATE	AREA	MONTH	UNIT_SAL	RETURNS
2	400-10-12	12 oz. Can	Smoothie Beer	New York	Manhattan	Feb	32	0
3	400-10-32	1 Liter Bottle	Smoothie Beer	New York	Manhattan	Feb	81	0
4	400-10-12	12 oz. Can	Smoothie Beer	New York	Bronx	Feb	91	2
5	400-10-32	1 Liter Bottle	Smoothie Beer	New York	Bronx	Feb	3	0
6	400-20-12	12 oz. Can	Diet Smoothie	New York	Manhattan	Feb	57	1
7	400-20-32	1 Liter Bottle	Diet Smoothie	New York	Manhattan	Feb	94	2
8	400-20-12	12 oz. Can	Diet Smoothie	New York	Bronx	Feb	25	0
9	400-20-32	1 Liter Bottle	Diet Smoothie	New York	Bronx	Feb	78	1
10								

**Note:** If the output sheet name is already open before the drill action, then a new file is created for the query results.



---

## Requirement 2

When performing variance analysis between actual and budget data, users occasionally need to view more detailed information at weekly level. Weekly data is not stored in the Essbase database. On a dimension-by-dimension basis, users require the following:

- **Product Dimension.** When users view information about product families or product groups, transactions by product group must be returned. The retrieved records must match the product code of the product currently being viewed in the Essbase model.
- **Market Dimension.** Users want to view transactional data only when they are looking at regions or states in the Market dimension. When users view regional data from Essbase, the states belonging to that region are retrieved. When users view state data from Essbase, the relational records for the state are retrieved.
- **Year Dimension.** Users want to view weekly data only when they are looking at the lowest level in the Year dimension. In this case, users want to view weekly data that relates to the month they are currently viewing from Essbase.
- **Scenario Dimension.** The Scenario data in Essbase is irrelevant because users always want to view actual figures from the relational database.
- **Measures Dimension.** The Measures data in Essbase is irrelevant because users always want to view unit sales from the relational database.

To further define the drill-through requirement, the user shows the following spreadsheets and describes the desired behavior:

	A	B	C	D	E	F	G
1	Sales						
2	New York						
3			Actual			Variance	
4		Jan	Feb	Mar	Jan	Feb	Mar
5	100	135	187	116	(25)	(33)	(14)
6	200	162	241	68	(8)	(9)	(2)
7	300	235	220	271	(6)	(10)	1
8	400	50	232	309	6	(8)	(11)
9	Product	588	880	764	(32)	(50)	(26)

If the user selects cell C8 (the data cell having attributes of Sales, New York, Actual, Feb, 400), the user wants to see all transactions that are within the New York area, within product family 400, and for all weeks in the month of February.

	A	B	C	D	E	F	G
1	Sales						
2	New York						
3			Actual			Variance	
4		Jan	Feb	Mar	Jan	Feb	Mar
5	400-10	234	232	234	(16)	(18)	(16)
6	400-20	219	243	213	(11)	(17)	(17)
7	400-30	134	189	198	(6)	(11)	(12)
8	400	587	664	645	(33)	(45)	(45)
9	Product	2479	2625	2601	179	175	161

If the user selects cell C6 (which has data cell attributes of Sales, New York, Actual, Feb, 400-20), the user wants to see all transactions that are within the New York area, within product group 400-20, and for all weeks in the month of February.

You need the WEEKLY, CALENDAR, and TERRITOR tables, described earlier in this chapter, to fulfill the user's requirement. WEEKLY already has most of the elements required to form the query and return the appropriate data. However, to map the month-to-week relationship, the CALENDAR table must be joined to the WEEKLY table.

The WEEKLY table has a column for the state in which products are sold. The user requirements, however, are for an SQL Drill-Through to occur from either a region or state member. The TERRITOR table contains region and state relationships, so you need to join TERRITOR to WEEKLY to perform the required query.

In effect the user requires access to a view with the following columns: PROD\_FAM, MONTH, W\_ENDING, REGION, STATE, and UNIT\_SALES. The resultant view would appear as follows:

PROD_FAM	MONTH	W_ENDING	REGION	STATE	UNIT_SALES
100-	Jan-95	1/8/95	East	New York	86
100-	Jan-95	1/15/95	East	New York	20
100-	Jan-95	1/22/95	East	New York	25
100-	Jan-95	1/29/95	East	New York	28
100-	Feb-95	2/5/95	East	New York	34

To define the SQL Drill-Through profile, do the following:

1. Start an Excel or 1-2-3 session, and connect to the Sample Basic database.
2. Open SQLDRILL.XLS or SQLDRILL.WK4 from the \ESSBASE\CLIENT\SAMPLE directory on the drive in which you installed the Essbase Spreadsheet Add-in.
3. Select cell C8 (the data cell having attributes of Sales, New York, Actual, Feb, 400) and choose Essbase | SQL Drill-Through.

The **SQL Database Login** dialog box appears.

4. Click **Query Options**.

Essbase attempts to connect to the data source. If successful, the **Query Options** dialog box appears.

5. Click **Add Profile**.
6. Clear the **Use current profile values as defaults** check box, enter **Weekly Details** as the name, and click **OK**.
7. Select **Weekly Details** in the **Output Profile** list box, then click **Edit Profile**.

The **Profile Editor** dialog box appears:



8. Switch to the **Defined Columns** page of the **Profile Editor** dialog box. This lets you define the columns available to the user from one or more relational tables or views.



The **Tables in Database** list box shows a list of all tables defined for the data source. In the case of dBASE, all .DBF files located in the directory specified by the ODBC Administrator Utility are listed. The **[Define Table]** option in the **Tables in Database** list box lets you add tables to the list.

**Note:** When joining multiple dBASE tables, the SQL request fails if the dBASE table names include a path name.

9. To add the tables to the mapping profile:
  - a. Double-click **[Define Table]** in the **Tables in Database** list box.

The **Add New Table to Profile** dialog box appears:



- b. Enter `WEEKLY` as the name and click OK. Essbase looks for the `WEEKLY` table in the directory defined by the ODBC Administrator Utility. If you need to specify a table outside of the defined directory, you can provide a file path. However, specifying a file path causes a dBASE table join operation to fail.
    - c. Double-click **[Define Table]**.
    - d. Enter `CALENDAR` as the name and click OK.
    - e. Double-click **[Define Table]**.
    - f. Enter `TERRITOR` as the name and click OK.

Using **[Define Table]** has three advantages:

- It provides an alternate way to copy table names from the **Tables in Database** list box to the **Tables in profile** list box.
- It lets you add a table name that doesn't exist yet, but will exist when the drill action occurs.
- It lets you add a table name without a path—a requirement for linked dBASE tables.

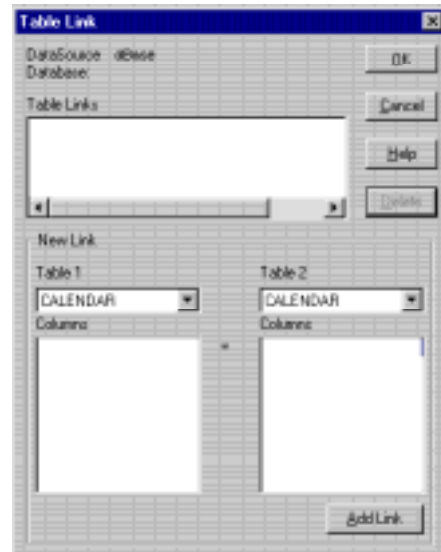
The **Tables in profile** list box in the **Defined Columns** page of the **Profile Editor** dialog box then appears as follows:



To return all columns required by the user, the tables must be joined.

10. Click **Table Link**. This button is available only if more than one table name appears in the **Tables in profile** list box.

The **Table Link** dialog box appears:

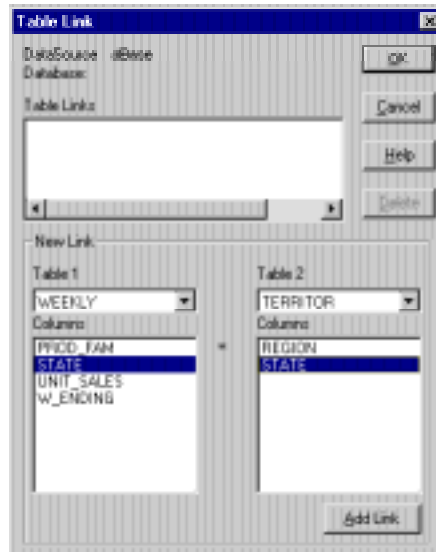


**Note:** To perform a table link with dBASE tables, the tables must have corresponding index files. Index files for the sample dBASE tables are provided with the SQL Interface installation. For more information on linking tables, see the *SQL Interface Guide*.



The first link will join the WEEKLY and TERRITOR tables, based on the STATE columns in each table.

- a. Choose WEEKLY from the **Table 1** list box, then choose the STATE column.
- b. Choose TERRITOR from the **Table 2** list box, then choose the STATE column.

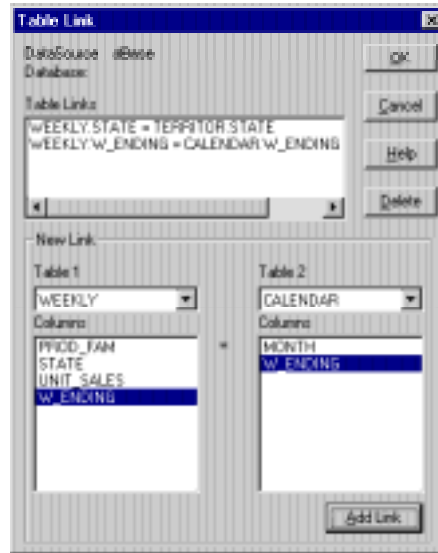


- c. Click **Add Link** to define the link. The link definition appears in the **Table Link** list box: WEEKLY . STATE=TERRITOR . STATE

The second link will join the WEEKLY and CALENDAR tables, based on the W\_ENDING columns in each table.

- a. Choose WEEKLY from the **Table 1** list box, then choose the W\_ENDING column.
- b. Choose CALENDAR from the **Table 2** list box, then choose the W\_ENDING column.

- c. Click **Add Link** to define the link. The link definition appears in the **Table Link** list box.



- d. Click **OK** to close the **Table Link** dialog box.

**Note:**

If you click **Cancel** on the **Defined Columns** page of the **Profile Editor** dialog box, any links you created are not saved.

11. Add columns to the **Defined Column List** as follows:
- Choose the **WEEKLY** table. Choose the **PROD\_FAM** column and click **Copy->**.
  - Choose the **CALENDAR** table. Choose the **MONTH** column and click **Copy->**. Choose the **W\_ENDING** column and click **Copy->**.
  - Choose the **TERRITOR** table. Choose the **REGION** column and click **Copy->**.
  - Choose the **WEEKLY** table. Choose the **STATE** column and click **Copy->**. Choose the **UNIT\_SALES** column and click **Copy->**.

The **Defined Column List** in the **Defined Columns** page of the **Profile Editor** dialog box appears as follows:



12. Once the columns are defined, switch to the **SQL Generator** page of the **Profile Editor** dialog box to define the mapping rules for each dimension.

For the Market dimension, the SQL Drill-Through should occur only if the second or third generation number of the Market dimension is attained. If you are viewing data based on a region, then a Where clause to the REGION column of the TERRITOR table is required. If you are viewing data based on a state, you need a Where clause to the STATE column.

To define the mapping rules for Market:

- a. Choose **Market** from the **SQL Rules for** list box.
- b. Check **User must attain at least generation**, and enter a value of 2.
- c. Choose **Don't allow SQL Drill-Through**.

d. Define five new SQL generation rules as follows:

<b>If Market Matches...</b>	<b>Add This Where Clause...</b>
Central	TERRITOR.REGION='Central'
South	TERRITOR.REGION='South'
West	TERRITOR.REGION='West'
East	TERRITOR.REGION='East'
* (Pattern)	TERRITOR.STATE='\1'

As you create each new rule, it appears after the currently selected rule in the **SQL Generation Rules** list box. The order in which these rules appear in the list is important because the wildcard match is used only if none of the first four rules are met. This occurs when the user is viewing data based on a state from the Market dimension.

Once all the rules are entered, they should appear in the **SQL Generation Rules** list box as follows:

```

Central:TERRITOR.REGION='Central'
South:TERRITOR.REGION='South'
West:TERRITOR.REGION='West'
East:TERRITOR.REGION='East'
*:TERRITOR.STATE='\1'

```

For the Product dimension, the SQL Drill-Through should occur only if the product-family generation or deeper is attained. In the Sample Basic database, the product-family generation starts at generation 2.

To define an SQL rule for Product:

- Choose **Product** from the **SQL Rules for** list box.
- Check **User must attain at least generation**, and enter a value of 2.
- Choose **Don't allow SQL Drill-Through**.
- Define three new SQL generation rules as follows:

<b>If Product Matches...</b>	<b>Add This Where Clause...</b>
???-?? (Pattern)	WEEKLY.PROD_FAM LIKE '\1-\2-%'
Diet	WEEKLY.PROD_FAM LIKE '%-20'
??? (Pattern)	WEEKLY.PROD_FAM LIKE '\1-%'

Once all the rules are entered, they should appear in the **SQL Generation Rules** list box as follows:

```

???-??:WEEKLY.PROD_FAM LIKE '\1-\2-%'
Diet:WEEKLY.PROD_FAM LIKE '%-20'
???-??:WEEKLY.PROD_FAM LIKE '\1-%'

```

For the Year dimension, the SQL Drill-Through should occur only if the generation with the monthly data is attained. In the Sample Basic database, the monthly generation is number 3.

To define an SQL rule for Year:

- Choose **Year** from the **SQL Rules for** list box.
- Check **User must attain at least generation**, and enter a value of 3.
- Choose **Don't allow SQL Drill-Through**.
- Define an SQL generation rule. MONTH column values from the dBASE tables are in the form of Mon-YY. Define the rule as follows:

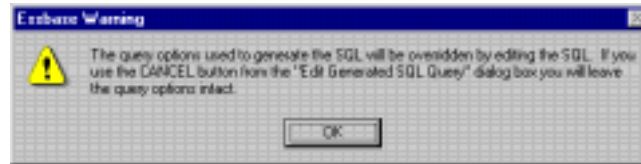


The rule is based on a 3-digit wildcard pattern to match a month name from the Year dimension. For example, if the active data cell has an attribute of Jan, then the Where clause becomes `CALENDAR.MONTH = 'Jan-95'`.

Based on the user requirements, the Measures and Scenario dimensions do not require SQL Drill-Through mapping rules.

13. After the SQL rules are defined, click OK.
14. Close the **Query Options** dialog box to return to the **SQL Database Login** dialog box.
15. Choose **Edit SQL** to view the SQL statement created by the profile definition.

The following message appears:



16. Click OK.

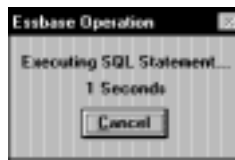
The **Edit Generated SQL Query** dialog box appears:



17. Click OK to return to the **SQL Database Login** dialog box.


18. Click **Drill**.

The following message appears:



The Essbase server passes the SQL statement to the source database. The query operation is asynchronous, so you can cancel it at any time. Once the query is completed on the source, data records are sent to the output destination.

The data appears as follows:

A screenshot of an Essbase data grid window titled "sqldata.xls". The grid displays a table with 11 rows and 7 columns. The columns are labeled: WEEKLY, CALENDAR, CALENDAR, TERRITOR, WEEKLY, WEEKLY, and UNIT SALES. The data rows contain values for each column, including dates, territories, and unit sales figures.

	WEEKLY	CALENDAR	CALENDAR	TERRITOR	WEEKLY	WEEKLY	UNIT SALES
2	400-10	Feb-95	1995-02-05	East	New York		43
3	400-10	Feb-95	1995-02-12	East	New York		25
4	400-10	Feb-95	1995-02-19	East	New York		80
5	400-10	Feb-95	1995-02-26	East	New York		67
6	400-20	Feb-95	1995-02-05	East	New York		15
7	400-20	Feb-95	1995-02-12	East	New York		29
8	400-20	Feb-95	1995-02-19	East	New York		4
9	400-20	Feb-95	1995-02-26	East	New York		1
10							
11							





---

# Appendix A

# The SQLDRILL.INI File

This appendix shows the `SQLDRILL.INI` that is shipped with the Arbor Essbase SQL-Drill Through installation. The settings in the `.INI` file are maintained through the SQL Drill-Through dialog boxes in the Arbor Essbase Spreadsheet Add-in.

**Note:** Use care when editing the `.INI` file. If you change any other setting in the file besides the `Supervisor` and `SQLDrillServer` settings, SQL Drill-Through may become unusable or may cause any profiles that are already defined to become inoperable. Make a backup copy of the `.INI` file before making any further modifications.

---

## The Supervisor Setting

The `Supervisor` setting is for database administrators and advanced users, and you must set it manually. It defines whether a user can create, edit, or delete mapping profiles. You can edit the `.INI` file using a text editor such as Notepad.

The `Supervisor` setting is in the `[Login]` section. It has the following values:

- 0 (zero): the user cannot use the **Profile Editor** dialog boxes to edit profile information stored in the `.INI` file. Use 0 for users who will simply use profiles that you create.
- 1: the user can create, edit, and delete profiles using the SQL Drill-Through dialog boxes in the Spreadsheet Add-in. Use 1 for database administrators and advanced users who need to modify profiles.

You need to set the `Supervisor` setting to 0 before distributing the `.INI` file to end users. To disable profile editing:

1. Open the `SQLDRILL.INI` file in any Windows editor such as Notepad. The file is located in the Windows directory.
2. Find the `Supervisor` setting. It is most likely set to 1. Change the setting to 0.
3. Save the `.INI` file.

---

## The SQLDrillServer Keyword

You can access your relational data from an ODBC driver through the Arbor Essbase OLAP server, or directly from an ODBC driver on your client machine. The `SQLDrillServer` keyword determines how your source data is accessed.

Here is the `SQLDrillServer` keyword syntax:

```
SQLDrillServer=n
```

Where *n* is 1 for Server Drill-Through and 0 for Client Drill-Through. The default is 1.

- Server SQL Drill-Through sends the client's SQL statement to the ODBC driver via the Essbase server.
- Client SQL Drill-Through sends the client's SQL statement directly to the ODBC driver, which must be installed on the client machine.

---

## The Default SQLDRILL.INI File

The default `.INI` that is shipped with the SQL Drill-Through installation is shown on the following pages. Note that any line which is preceded with a semicolon is interpreted as a comment line.

```

; SQLDRILL.INI - Contains default mapping profiles that are
;                 shipped with the SQL Drill-Through Module.
;
; Login Section, this stores persistent login information
;

[Login]
Supervisor=1
Security=0
DataSource=
DataSourceCurrent=dBase
ServerCurrent=
DatabaseCurrent=
DictName=
FileName=
Username=
Application=
SQLDrillServer=1

;
; This section defines the destination for SQL data. SQL drill
; data can go to one of 3 destinations: File, Printer, and a new
; Sheet. These destinations are given destination codes:
;     FILE           1
;     PRINTER        2
;     SHEET          3
; Other entries define the other output options

[Output]
Destination=3
IncludeColumnNames=1
LimitOutput=1
OutputLimit=10
Name1=C:\SQLDATA.TXT
Name2=
Name3=C:\SQLDATA.XLS

;
; User Exit Section:
; This section describes the user exit facility of the
; SQL drill down.
; A user DLL is specified along with function entry points and
; flags as to whether they are to be called or not
; fPre and fPost are the flags to decide whether or not to call
; the PreFunction and PostFunction respectively.
;

[UserExit]
UserDll=USREXIT.DLL
PreFunction=UserExitPreFn
PostFunction=UserExitPostFn
fPre=0
fPost=0

;;
;; Global Profile Section
;;
; The global profile section
; administers all profiles defined in the ini file.
;

```

```

; It contains names of all defined profiles, an indicator as to
; which one is current.
;

[Profile]
Entries=Sample1,Sample2
Current=Sample2
UseAsDefault=1

[Sample1.Tables]
Entries=area
Current=area

[Sample1.Columns]
Entries=PROD_CODE-,PACKAGE-,DESCRIPTIO-,
,STATE+,AREA+,MONTH+,UNIT_SALES+,RETURNS+
Current=RETURNS
AllowStar=1
StarFlag=1

[Sample1.Links]
Entries=
Current=

[Sample1.Rules]
Entries=Year,Measures,Scenario,Market,Product
Current=Product

[Sample1.Scope]
Entries=112,002,002,112,002
Current=002

[Sample1.Rules.Year]
1=Qtr1:MONTH='Jan' OR MONTH = 'Feb' OR MONTH = 'Mar'
2=Qtr2:MONTH='Apr' OR MONTH='May' OR MONTH='Jun'
3=Qtr4:MONTH='Oct' OR MONTH='Nov' OR MONTH='Dec'
4=Qtr3:MONTH='Jul' OR MONTH='Aug' OR MONTH='Sep'
5=???:MONTH='\1'

[Sample1.Rules.Market]
1=*:STATE='\1'

[Sample1.Rules.Product]
1=???-?:PROD_CODE LIKE '\1-\2-%'
2=Diet:PROD_CODE LIKE '%-20'
3=???:PROD_CODE LIKE '\1-%'

[Sample2.Rules]
Entries=Year,Measures,Scenario,Market,Product
Current=Product

[Sample2.Rules.Year]
1=???:CALENDAR.MONTH='\1-95'

[Sample2.Rules.Market]
1=Central:TERRITOR.REGION='Central'
2=South:TERRITOR.REGION='South'
3=West:TERRITOR.REGION='West'
4=East:TERRITOR.REGION='East'
5=*:TERRITOR.STATE='\1'

```

```
[Sample2.Rules.Product]
1=???-?:WEEKLY.PROD_FAM LIKE '\1-\2-%'
2=Diet:WEEKLY.PROD_FAM LIKE '%-20'
3=????:WEEKLY.PROD_FAM LIKE '\1-%'

[Sample2.Scope]
Entries=113,002,002,112,002
Current=002

[Sample2.Tables]
Entries=calendar,territor,weekly
Current=weekly

[Sample2.Links]
Entries=weekly.STATE = territor.STATE,weekly.W_ENDING =
calendar.W_ENDING
Current=weekly.STATE = territor.STATE

[Sample2.Columns]
Entries=calendar.MONTH+,weekly.W_ENDING+,territor.REGION+,weekly.
STATE+,weekly.PROD_FAM+,weekly.UNIT_SALES+
Current=weekly.UNIT_SALES
AllowStar=1
StarFlag=0
```



This appendix describes how you can add your custom functions to the Arbor Essbase SQL Drill-Through product. As an application developer, you can create your own SQL functions in a single-user DLL, tie them into the SQL Drill-Through product, and have them called by SQL Drill-Through during the drill execution.

To add custom functions to SQL Drill-Through, open the **Profile Editor** dialog box and switch to the **User Exit** page.

The pre-SQL function is called prior to making the SQL connection. The function must match the following prototype:

```
extern "C" BOOL _far _pascal _export fnSQLPreFunction (LPSTR szSelect,
                                                    LPSTR szFrom,
                                                    LPSTR szWhere);
```

The LPSTR parameters represent the Select, From, and Where clauses from the SQL query that will be executed by SQL Drill-Through. The Boolean returned should be FALSE (0) if you want SQL Drill-Through to continue with the query execution, or TRUE (non-zero) if you want to end the query.

The post-SQL function is called after the SQL query has been processed. The function must match the following prototype:

```
extern "C" BOOL _far _pascal _export fnSQLPostFunction ();
```

Note that there are no parameters passed to the post-SQL function.

The post-SQL function is called regardless of the return value from the pre-SQL function. Additionally, although the post-SQL function returns a Boolean, the current implementation of SQL-Drill Through doesn't act upon this value.

The following examples show when the functions are useful:

- If the user wants to analyze the query and create a local slice of their larger database against which the query could be executed, this could be done in the pre-SQL function. Deletion and any other clean-up of the locally created tables could be done in the post-SQL function. In this case, the pre-SQL function would return FALSE so that SQL Drill-Through performs the actual query.

- If the user needs to execute the SQL query themselves, perhaps making modification prior to execution. This could be done in the pre-SQL function, and the user would return `TRUE` to stop the query execution in SQL Drill-Through. Note that this would make the user completely responsible for query output handling.

For more information about the **User Exit** page of the **Profile Editor** dialog box, click **Help**.