



Q u i c k   a n s w e r s   t o   c o m m o n   p r o b l e m s

# Oracle Essbase 11 Development Cookbook

Over 90 advanced development recipes to build and take your  
Oracle Essbase Applications further

**Jose R. Ruiz**

**[PACKT]** enterprise   
PUBLISHING professional expertise distilled

# Oracle Essbase 11 Development Cookbook

Over 90 advanced development recipes to build and take  
your Oracle Essbase Applications further

**Jose R. Ruiz**

**[PACKT]** enterprise   
PUBLISHING professional expertise distilled

BIRMINGHAM - MUMBAI

# Oracle Essbase 11 Development Cookbook

Copyright © 2012 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing, and its dealers and distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

First published: January 2012

Production Reference: 1170112

Published by Packt Publishing Ltd.  
Livery Place  
35 Livery Street  
Birmingham B3 2PB, UK.

ISBN 978-1-84968-326-5

[www.packtpub.com](http://www.packtpub.com)

Cover Image by Sandeep Babu ([sandyjb@gmail.com](mailto:sandyjb@gmail.com))

# Credits

**Author**

Jose R. Ruiz

**Copy Editor**

Neha Shetty

**Reviewers**

Alexia Rodriguez Alwine

Satyanarayana Bodhanapu

**Project Coordinator**

Vishal Bodwani

**Acquisition Editor**

Kerry George

**Proofreaders**

Aaron Nash

Chris Smith

**Lead Technical Editor**

Susmita Panda

**Indexer**

Rekha Nair

**Technical Editor**

Llewellyn F. Rozario

**Production Coordinator**

Arvindkumar Gupta

**Cover Work**

Arvindkumar Gupta

# About the Author

**Jose R. Ruiz** is an Oracle Essbase 11 Certified Implementation Specialist with over nine years experience in developing enterprise-level Essbase applications. He has maintained and conducted post-production development on 18 Essbase databases. In addition, Jose Ruiz has been charged with developing E-commerce, Fixed Assets, Balance Sheets, Point of Sales, and Inventory databases.

Jose Ruiz is currently working with Oracle consultants on designing, developing, and implementing an Inventory, Purchase Order, and Sales Data Mart and an Essbase database at his current employer.

---

I would like to thank my colleagues and friends Peter Beddoe and Alexia Alwine for their review and advice. In addition, I would like to thank my wife, Yaneth C. Ruiz, for her support and patience throughout this endeavor.

---

# About the Reviewer

**Alexia Rodriguez Alwine** is a Project Manager with extensive experience in the pharmaceutical and consumer products industries. She has worked with Unilever, Inc. and several of its subsidiaries; Steifel Laboratories, a GlaxoSmithKline company; and BE Aerospace. In addition to serving as a Project Manager, she has served as Hyperion Administrator, Systems Analyst, and Finance Manager. Her experience with Oracle includes Web Analysis, Financial Reporting, FDM, HFM, Hyperion Planning, Oracle Upgrades, and Essbase Migration Projects. She also has experience with SAP and Data Mart implementation.

Alexia graduated with a bachelor's degree in economics, communications, and international relations from the University of Pennsylvania. She received her MBA from the University of Florida. In her spare time, she researches and conducts workshops concerning the impact of technology on the family.

# www.PacktPub.com

## Support files, eBooks, discount offers, and more

You might want to visit [www.PacktPub.com](http://www.PacktPub.com) for support files and downloads related to your book.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at [www.PacktPub.com](http://www.PacktPub.com) and, as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at [service@packtpub.com](mailto:service@packtpub.com) for more details.

At [www.PacktPub.com](http://www.PacktPub.com), you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.



<http://PacktLib.PacktPub.com>

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can access, read, and search across Packt's entire library of books.

## Why Subscribe?

- ▶ Fully searchable across every book published by Packt
- ▶ Copy and paste, print, and bookmark content
- ▶ On demand and accessible via web browser

## Free Access for Packt account holders

If you have an account with Packt at [www.PacktPub.com](http://www.PacktPub.com), you can use this to access PacktLib today and view nine entirely free books. Simply use your login credentials for immediate access.

## Instant Updates on New Packt Books

Get notified! Find out when new books are published by following [@PacktEnterprise](#) on Twitter, or the *Packt Enterprise* Facebook page.

# Table of Contents

<b>Preface</b>	<b>1</b>
<b>Chapter 1: Understanding and Modifying Data Sources</b>	<b>7</b>
Introduction	8
Setting up an Account or Measures dimension with a parent-child reference	8
Setting up dimensions with a generation reference	16
Adding columns for outline formulas	18
Adding the solve order column to tables that have ASO formulas	20
Adding and populating the Sort Order Column	22
Adding tables for varying attributes	26
Determining hierarchies in relational tables	29
Using the Essbase Outline Extractor to extract dimensions	36
Using Star Analytics to build your star schema from existing Essbase cubes	40
<b>Chapter 2: Using Essbase Studio</b>	<b>45</b>
Introduction	45
Creating TBC sample database and connecting to the data source	46
Adding user-defined tables	51
Building your minischema	53
Setting up joins in a minischema	57
Adding tables to a minischema	60
Using a text file data source	63
Working with Common Platform Language (CPL)	67
Using Sort Order on data elements	71
<b>Chapter 3: Building the BSO Cube</b>	<b>73</b>
Introduction	73
Creating hierarchies using a parent-child reference table	74
Creating hierarchies using a generation reference table	78



<b>Adding attribute dimensions to hierarchies</b>	<b>80</b>
<b>Building a Calendar dimension</b>	<b>83</b>
<b>Creating date elements</b>	<b>88</b>
<b>Creating Alias tables</b>	<b>91</b>
<b>Developing cube schema and an Essbase model</b>	<b>94</b>
<b>Setting Essbase properties</b>	<b>98</b>
<b>Deploying a cube</b>	<b>101</b>
<b>Creating an OLAP Model in EIS</b>	<b>104</b>
<b>Creating an OLAP metaoutline in EIS</b>	<b>111</b>
<b>Chapter 4: Building the ASO Cube</b>	<b>119</b>
Introduction	119
Using the Connection Wizard to set up an ASO cube	120
Building a Measures dimension from the fact table	123
Creating an ASO Cube Schema and an Essbase Model	126
Understanding Essbase Model properties for the ASO cube	129
Designing a drill-through report	132
Using the View dimension for Dynamic Time Series reporting	136
<b>Chapter 5: Using EAS for Development</b>	<b>139</b>
Introduction	140
Adding an application and database on an Essbase Server	140
Using the outline editor to add dimensions	143
Using dimension build rules to add the parent-child dimension	147
Creating dimension build rules to add a base and attribute dimensions	151
Using dimension build rules to add user-defined attributes and associate dimensions	156
Creating load rules for flat file data loads	161
Creating substitution variables	165
Using If/Else logic and substitution variables in outline formulas	167
Using Text measures on a BSO cube	171
Using Date measures on a BSO cube	176
Using different outline formula logic at parent level	179
Creating a load rule for SQL data load using substitution variables	181
Using MDX in aggregate storage applications	186
<b>Chapter 6: Creating Calculation Scripts</b>	<b>191</b>
Introduction	191
Using Essbase Set function commands and Calc All to calculate cubes	192
Using control flow commands, conditional, and logical operators	196
Using substitution variables in calculations script	201
Using UDAs and Calc Two Pass in calculation scripts	204

Using Attributes in calculation scripts	210
Clearing data and using the cross- dimensional operators in a calculation script	215
Using allocation functions in calculation scripts	219
Modifying Essbase settings to improve calculation performance	223
Using MDX to calculate Aggregate Storage database	228
<b>Chapter 7: Using MaxL to Automate Process</b>	<b>233</b>
Introduction	234
Setting up folder structure and other files needed for MaxL automation	234
Executing dimension build rules using MaxL	242
Executing load rules using MaxL	247
Executing calculations using MaxL	250
Executing partitions using MaxL	254
Executing report scripts using MaxL	260
Adding or changing substitution variables with MaxL	264
Using ASO incremental data loads	266
Using encryption in MaxL scripts	268
Deploy dimension created in Essbase Studio	271
<b>Chapter 8: Data Integration</b>	<b>273</b>
Introduction	273
Using report script to extract data to a text file	274
Using the DATAEXPORT function to extract data into a text file	279
Using the DATAEXPORT function to extract data into a relational source	283
Exporting data using column format	287
Using MaxL to extract the outline in XML format	291
Using @XREF functions to move data between BSO cubes	292
Partitioning data from BSO to ASO cubes	296
Using MDX for extracting data using API	303
There's more	311
<b>Chapter 9: Provisioning Security Using MaxL Editor or Shared Services</b>	<b>313</b>
Introduction	313
Using MaxL editor to add and externalize a user	314
Using Shared Services to add and provision a user	317
Using MaxL Editor to set up a filter for MetaRead and Write access	321
Using Shared Services to provision filters to a group	324
Using Shared Services to provision calculation scripts to a group	329
Using MaxL to export security file	333

<b>Chapter 10: Developing Dynamic Reports</b>	<b>335</b>
<b>Introduction</b>	<b>335</b>
<b>Creating a connection and using substitution variables in financial reports</b>	<b>336</b>
<b>Using the column templates and formatting reports</b>	<b>341</b>
<b>Retrieving data using UDAs and Attributes</b>	<b>348</b>
<b>Retrieving data using children and descendants member set functions</b>	<b>352</b>
<b>Using User Prompts and the POV to select members</b>	<b>356</b>
<b>Using conditional formatting and suppression in financial reports</b>	<b>360</b>
<b>Adding related content to financial reports</b>	<b>366</b>
<b>Creating a web analysis report</b>	<b>370</b>
<b>Index</b>	<b>377</b>

# Preface

*Oracle Essbase 11 Development Cookbook* will help you learn the tools necessary for the development of Essbase databases in Oracle Essbase version 11.1.2.1. Here you will find over 90 recipes that explain everything from how to use a relational data model to building and loading an Essbase database in Essbase Studio. The book also goes over how to build the Block Storage (BSO) databases and explains some of the options are exclusive to building an Aggregate Storage (ASO) database. In this book, we will be using Essbase Studio, Essbase Integration Services (EIS), and Essbase Administration Service (EAS) to build databases, and we will discuss the strengths of each tool. Moreover, we discuss how to create Calculation Scripts, use MaxL to automate your processes, and integrate data. Finally, we step through how to effectively implement security, and how to build dynamic reports. The reader is encouraged to use these recipes as the foundation for their own customized databases and scripts.

## What this book covers

*Chapter 1, Understanding and Modifying Data Sources.* This chapter explains how to prepare your data source to build hierarchies and load data in Essbase databases. Because you should not have to rebuild the wheel, we cover some tools that will assist us in extracting hierarchies from existing Essbase databases for the purpose of setting up your star schema in a relational environment. The goal of this chapter is to show the reader the components needed to maintain metadata in a relational environment and set up that environment to support drill-through reporting. This being said, most of the techniques used in this chapter can be implemented using flat files as well.

*Chapter 2, Using Essbase Studio.* We will begin this chapter by discussing advantages of and disadvantages of Essbase Studio when compared to development tools like Essbase Integration Services (EIS) and Essbase Administration Services (EAS). This chapter also has some of the more basic yet necessary steps needed to build your database using Essbase Studio. We will review how to create a data source, minischema, and manipulate data elements with Common Platform Language (CPL).

*Chapter 3, Building the BSO Cube.* In this chapter, we build and deploy the TBC Block Storage (BSO) database using Essbase Studio. We also explore the building of TBC databases using Essbase Integration Services (EIS).

*Chapter 4, Building the ASO Cube.* This chapter explains some of the options exclusive to building the Aggregate Storage (ASO) model. In addition, we learn how to build a Measure dimension from the fact table, and how to build a drill-through report in Essbase Studio.

*Chapter 5, Using EAS for Development.* This chapter explains how to build the Sample Basic database using the Essbase Administration Services (EAS) outline editor, build rules, load rules, and flat files. We also explore the use of Text and Date measures, outline formulas in the BSO model, and MDX in an aggregate storage database.

*Chapter 6, Creating Calculation Scripts.* In this chapter, we learn how to use calculation scripts to run complex formulas that require multiple passes through the Essbase database, data allocations, copying data, clearing data, aggregating data, and some best practices for optimizing your calculations' performance.

*Chapter 7, Using MaxL to Automate Process.* This chapter teaches you how to automate the updating, building, and loading of an Essbase database. This chapter more specifically shows MaxL script techniques designed to make scripts reusable and portable. These techniques will allow us to move our automation from development to staging or production without having to re-write our MaxL script before migration.

*Chapter 8, Data Integration.* This chapter explains how to integrate data in between Essbase and relational databases. In addition, we discuss how to move data between Essbase databases.

*Chapter 9, Provisioning Security using MaxL Editor or Shared Services.* This chapter shows how to use Shared Services and MaxL to set up security. Essbase has very flexible and powerful security features. This functionality, if planned carefully, can make your database more intuitive and customized to the needs of each end user.

*Chapter 10, Developing Dynamic Reports.* In this chapter, you will learn how to build a more dynamic Financial Report. Moreover, we discuss how to build a simple Web Analysis Report for an even more dynamic user experience.

## **What you need for this book**

You will need the following software to complete the recipes in this book:

1. Oracle EPM Essbase 11.1.2.1
2. Essbase Studio 11.1.2.1
3. Essbase Integration Services (EIS)
4. Financial Report & Web Analysis

5. SQL Server 2008/ Oracle 11g
6. Essbase Outline Extractor
7. Star Integration Server – Express Edition

## Who this book is for

If you are an experienced Essbase developer, Essbase Database Designer or Database Administrator, then this book is for you. This book assumes that you have good knowledge of Oracle Essbase.

## Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

Code words in text are shown as follows: "enter connection: \'TBC\'::\'TBC.dbo.MEASURES\'.'CHILD\'|\' - \'|\'connection : \'TBC\'::\'TBC.dbo.MEASURES\'.'MEASURES\_ALIAS\' in the textbox."

A block of code is set as follows:

```
Create Table PRODUCTS(  
    PRODUCTID    int          NOT NULL,  
    SKU           varchar(15)  NULL,  
    SKU_ALIAS     varchar(25)  NULL,  
    Constraint PK_PRODUCTS_PRODUCTID Primary Key (PRODUCTID)  
);
```

New terms and important words are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "Click on cell **F2**, then click on the box to the right and bottom of the cell, and drag it down to cell **F12**."



Warnings or important notes appear in a box like this.



Tips and tricks appear like this.

## Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to [feedback@packtpub.com](mailto:feedback@packtpub.com), and mention the book title via the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on [www.packtpub.com/authors](http://www.packtpub.com/authors).

## Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

## Downloading the example code

You can download the example code files for all Packt books you have purchased from your account at <http://www.PacktPub.com>. If you purchased this book elsewhere, you can visit <http://www.PacktPub.com/support> and register to have the files e-mailed directly to you.

## Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you would report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting <http://www.packtpub.com/support>, selecting your book, clicking on the **errata submission form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded on our website, or added to any list of existing errata, under the Errata section of that title. Any existing errata can be viewed by selecting your title from <http://www.packtpub.com/support>.

## **Piracy**

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at [copyright@packtpub.com](mailto:copyright@packtpub.com) with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

## **Questions**

You can contact us at [questions@packtpub.com](mailto:questions@packtpub.com) if you are having a problem with any aspect of the book, and we will do our best to address it.





# 1

## Understanding and Modifying Data Sources

In this chapter, we will cover the following topics:

- ▶ Setting up an Account or Measures dimension with a parent-child reference
- ▶ Setting up dimensions with a generation reference
- ▶ Adding columns for outline formulas
- ▶ Adding the Solve Order column to tables that have ASO formulas
- ▶ Adding and populating the Sort Order Column
- ▶ Adding tables for varying attributes
- ▶ Determining hierarchies in relational tables
- ▶ Using the Essbase Outline Extractor to extract dimensions
- ▶ Using Star Analytics to build your star schema from existing Essbase cubes

## Introduction

In this chapter, we will build components into our relational environment that will allow us to successfully build an **Essbase** database and facilitate drill-through reporting. Although we are discussing relational data sources, the properties, attributes, and concepts discussed in this chapter can be used to build hierarchies off data sources such as flat files for example. The techniques used here can be used in tools like Essbase Administrative Services, Essbase Integration Services, and Essbase Studio. This chapter also has recipes on the Essbase Outline Extractor and Star Analytics. These two tools allow us to extract hierarchies from existing Essbase cubes. We would use these tools to extract existing hierarchies or modify existing hierarchies to build all or parts of our star schema.

## Setting up an Account or Measures dimension with a parent-child reference

In this recipe, we will set up a relational table in a parent-child reference format. We will also review the type of properties that can go in each column and their definitions. The **Account** or **Measure** dimension is normally the most dynamic dimension in a financial database and it is recommended that you use the parent-child structure to build the dimension in a relational environment. The parent-child reference also allows ragged hierarchies without having to add columns to your tables when an additional level or generation is needed. We will also review an alternative method, which requires us to use the measures field in our fact table to build our Measure dimension.

### Getting ready

To get started, open your SQL Server Management Studio, and add a database called TBC. For this recipe, we are using T-SQL, but the PL/SQL equivalent will be provided where applicable. You should add a **SCHEMA** called TBC using tools such as **TOAD**, **SQL Developer**, or **Golden**, if you are using Oracle.

### How to do it...


1. Run the following scripts to create the `Measures` table. We can change the script below to PL/SQL by replacing `int` with `INTEGER` and `varchar()` with `VARCHAR2()`. A screenshot of the table follows the script:

```
--This is the syntax in T-SQL
create table MEASURES
(
    SORTKEY          int          not null,
    MEASURESID       int          not null,
    PARENT           varchar(85)  null ,
```

```

CHILD          varchar(85)          not null,
MEASURES_ALIAS varchar(85)          null   ,
CONSOLIDATION  varchar(85)          null   ,
TWOPASSCALC    varchar(85)          null   ,
STORAGE        varchar(85)          null   ,
VARIANCEREPORTING varchar(85)       null   ,
TIMEBALANCE    varchar(85)          null   ,
SKIP           varchar(85)          null   ,
UDA            varchar(85)          null   ,
FORMULA        varchar(255)         null   ,
COMMENT_ESSBASE varchar(85)          null   ,
constraint PK_MEASURES primary key (MEASURESID)
)
Go

```

MEASURES	
	SORTKEY
	MEASURESID
	PARENT
	CHILD
	MEASURES_ALIAS
	CONSOLIDATION
	TWOPASSCALC
	STORAGE
	VARIANCEREPORTING
	TIMEBALANCE
	SKIP
	UDA
	FORMULA
	COMMENT_ESSBASE



#### Downloading the example code

You can download the example code files for all Packt books you have purchased from your account at <http://www.PacktPub.com>. If you purchased this book elsewhere, you can visit <http://www.PacktPub.com/support> and register to have the files e-mailed directly to you.

2. Execute the following scripts to add the data to your table:

```

INSERT INTO MEASURES (SORTKEY,MEASURESID,PARENT,CHILD,MEASURES_
ALIAS,
CONSOLIDATION,TWOPASSCALC,STORAGE,VARIANCEREPORTING,
TIMEBALANCE,SKIP,UDA,FORMULA,COMMENT_ESSBASE)
VALUES (100,1,'Measures','Profit','','+',',',
'X','','','','','','','');

```

```

INSERT INTO MEASURES (SORTKEY,MEASURESID,PARENT,CHILD,MEASURES_
  ALIAS,
    CONSOLIDATION,TWOPASSCALC,STORAGE,VARIANCEREPORTING,
    TIMEBALANCE,SKIP,UDA,FORMULA,COMMENT_ESSBASE) VALUES
    (200,2,'Profit','Margin','','+', '', 'X', '', '', '', '', '');

INSERT INTO MEASURES (SORTKEY,MEASURESID,PARENT,CHILD,MEASURES_
  ALIAS,
    CONSOLIDATION,TWOPASSCALC,STORAGE,VARIANCEREPORTING,TIMEBALANCE,
    SKIP,UDA,FORMULA,COMMENT_ESSBASE) VALUES
    (300,3,'Margin','Sales','','+', '',
    '', '', '', '', '', '');

INSERT INTO MEASURES (SORTKEY,MEASURESID,PARENT,CHILD,ME
  ASURES_ALIAS,CONSOLIDATION,TWOPASSCALC,STORAGE,VARIANCE
  REPORTING,TIMEBALANCE,SKIP,UDA,FORMULA,COMMENT_ESSBASE)
  VALUES (400,4,'Margin','COGS','Cost of Goods Sold','-
  ', '', '', 'E', '', '', '', '');

INSERT INTO MEASURES (SORTKEY,MEASURESID,PARENT,CHILD,MEASURES_
  ALIAS,
    CONSOLIDATION,TWOPASSCALC,STORAGE,VARIANCEREPORTING,TIMEBALANCE,
    SKIP,UDA,FORMULA,COMMENT_ESSBASE) VALUES
    (500,5,'Profit','Total Expenses', '', '-
    ', '', 'X', 'E', '', '', '', '');

INSERT INTO MEASURES (SORTKEY,MEASURESID,PARENT,CHILD,MEASURES_
  ALIAS,
    CONSOLIDATION,TWOPASSCALC,STORAGE,VARIANCEREPORTING,TIMEBALANCE,
    SKIP,UDA,FORMULA,COMMENT_ESSBASE) VALUES
    (600,6,'Total Expenses','Marketing', '', '+',
    '', '', 'E', '', '', '', '');

INSERT INTO MEASURES (SORTKEY,MEASURESID,PARENT,CHILD,MEASURES_
  ALIAS,
    CONSOLIDATION,TWOPASSCALC,STORAGE,VARIANCEREPORTING,TIMEBALANCE,
    SKIP,UDA,FORMULA,COMMENT_ESSBASE) VALUES
    (700,7,'Total Expenses','Payroll', '', '+', '',
    'E', '', '', '', '');

INSERT INTO MEASURES (SORTKEY,MEASURESID,PARENT,CHILD,MEASURES_
  ALIAS,
    CONSOLIDATION,TWOPASSCALC,STORAGE,VARIANCEREPORTING,
    TIMEBALANCE,SKIP,UDA,FORMULA,COMMENT_ESSBASE) VALUES
    (800,8,'Total Expenses','Misc','Miscellaneous', '+',

```

```

    ','','E',' ',' ',' ',' ',' ');

INSERT INTO MEASURES (SORTKEY,MEASURESID,PARENT,CHILD,MEASURES_
ALIAS,
    CONSOLIDATION,TWOPASSCALC,STORAGE,VARIANCEREPORTING,TIMEBALANCE,
    SKIP,UDA,FORMULA,COMMENT_ESSBASE) VALUES
    (900,9,'Measures','Inventory',' ','~',' ','O',' ',' ',' ',' ',' ');

INSERT INTO MEASURES (SORTKEY,MEASURESID,PARENT,CHILD,MEASURES_
ALIAS,
    CONSOLIDATION,TWOPASSCALC,STORAGE,VARIANCEREPORTING,
    TIMEBALANCE,SKIP,UDA,FORMULA,COMMENT_ESSBASE) VALUES
    (1000,10,'Inventory','Opening
    Inventory',' ','+',',' ','E','F',' ',' ',
    'IF(NOT @ISMBR(Jan)) "Opening Inventory"=@PRIOR("Ending
    Inventory");ENDIF;"Ending Inventory"="Opening
    Inventory"+Additions-Sales;',' ');

INSERT INTO MEASURES (SORTKEY,MEASURESID,PARENT,CHILD,MEASURES_ALI
AS,CONSOLIDATION,TWOPASSCALC,STORAGE,VARIANCEREPORTING,TIMEBALANCE
,SKIP,UDA,FORMULA,
COMMENT_ESSBASE) VALUES (1100,11,'Inventory','Additions',' ','~',' ',
',' ','E',' ',' ',' ',' ');

INSERT INTO MEASURES (SORTKEY,MEASURESID,PARENT,CHILD,MEASURES_
ALIAS,
    CONSOLIDATION,TWOPASSCALC,STORAGE,VARIANCEREPORTING,
    TIMEBALANCE,SKIP,UDA,FORMULA, COMMENT_ESSBASE) VALUES
    (1200,12,'Inventory','Ending
    Inventory',' ','~',' ',' ','E','L',' ',' ',' ');

INSERT INTO MEASURES (SORTKEY,MEASURESID,PARENT,CHILD,MEASURES_ALI
AS,CONSOLIDATION,TWOPASSCALC,STORAGE,VARIANCEREPORTING,TIMEBALANCE
,SKIP,UDA,FORMULA,
COMMENT_ESSBASE) VALUES (1300,13,'Measures','Ratios',' ','~',' ',
'O',' ',' ',' ',' ',' ');

INSERT INTO MEASURES (SORTKEY,MEASURESID,PARENT,CHILD,MEASURES_
ALIAS,
    CONSOLIDATION,TWOPASSCALC,STORAGE,VARIANCEREPORTING,TIMEBALANCE,
    SKIP,UDA,FORMULA, COMMENT_ESSBASE) VALUES
    (1400,14,'Ratios','Margin %',' ','+', 'T','X',' ',' ',' ',
    'Margin % Sales;',' ');

INSERT INTO MEASURES (SORTKEY,MEASURESID,PARENT,CHILD,MEASURES_
ALIAS,

```

```

CONSOLIDATION, TWOPASSCALC, STORAGE, VARIANCEREPORTING, TIMEBALANCE,
SKIP, UDA, FORMULA, COMMENT_ESSBASE) VALUES
(1500, 15, 'Ratios', 'Profit %', '~', 'T', 'X', '', '', '', '',
'Profit % Sales;', '');

INSERT INTO MEASURES (SORTKEY, MEASURESID, PARENT, CHILD, MEASURES_
ALIAS,
CONSOLIDATION, TWOPASSCALC, STORAGE, VARIANCEREPORTING, TIMEBALANCE,
SKIP, UDA, FORMULA, COMMENT_ESSBASE) VALUES
(1600, 16, 'Ratios', 'Profit per Ounce', '~', 'T', 'X', '', '', '', '',
'Profit/@ATTRIBUTEVAL (Ounces);', '');

```

## How it works...

The MEASURES table has the following columns:

COLUMN	DESCRIPTION
SORTKEY	This column is the integer that helps you sort the MEASURES in the order that you want them to appear in the hierarchy
MEASURESID	This ID is used as the PRIMARY KEY in the MEASURES table and as a FOREIGN KEY in the fact table
PARENT	This column is the Parent in the hierarchy
CHILD	This column is the Child of the Parent column
MEASURES_ALIAS	This is a more intuitive description of Measures normally defined by the business
CONSOLIDATION	This field has the aggregation type for the Child column
TWOPASSCALC	This field has the value "T" if the aggregation requires a second pass through the outline for the results to be right
STORAGE	Storage can have many values and will determine how or if the data in the outline is stored or dynamically calculated
VARIANCEREPORTING	The Variance Reporting column is used to mark Expense accounts for reporting variances
TIMEBALANCE	The Time Balance column is used with your time dimension to determine whether to use LIFO, FIFO, or the Average method for a specific measure
SKIP	The Skip column works with Time Balance to determine how to treat #MISSING or Zero values
UDA	The User Defined Attribute is useful for many purposes including outline formulas, calculation formulas, and the retrieval of data by the criteria defined by the business
FORMULA	These are the outline formulas used in the BSO model
COMMENT_ESSBASE	These are simply comments on the meta-data stored in this table

In step 2, we load the data. The following are descriptions of what goes into some of these columns as per Oracle's documentation.

These are the valid **Consolidations** values:

TYPE	TYPE DESCRIPTION	TYPE LONG DESCRIPTION
%	Percent	Expresses as a percentage of the current total in a consolidation
*	Multiplication	Multiplies by the current total in a consolidation
+	Addition	Adds to the current total in a consolidation
-	Subtraction	Subtracts from the current total in a consolidation
/	Division	Divides by the current total in a consolidation
^	Never	Excludes from all consolidations in all dimensions
~	Ignore	Excludes from the consolidation

This is the valid **Two Pass** value:

TYPE	TYPE DESC	TYPE LONG DESCRIPTION
T	Two Pass Calculation	Requires a two-pass calculation (applies to accounts dimensions only)

These are the valid **Storage** values:

TYPE	TYPE DESC	TYPE LONG DESCRIPTION
N	Never Share	Never allows data sharing
O	Label Only	Tags as label only (store no data)
S	Store Data	Sets member as stored member (non-Dynamic Calc and not label only)
V	Dynamic Calc and Store	Creates as Dynamic Calc and Store
X	Dynamic Calc	Creates as Dynamic Calc

This is the valid **Variance Reporting** value:

TYPE	TYPE DESC	TYPE LONG DESCRIPTION
E	Expense	Treats as an expense item (applies to accounts dimensions only)

These are the valid **Time Balance** values:



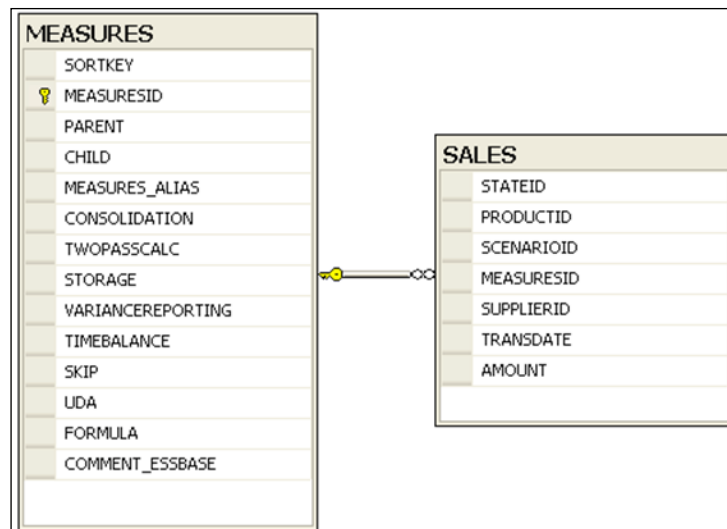
TYPE	TYPE DESC	TYPE LONG DESCRIPTION
A	Average	Treats as an average time balance item (applies to accounts dimensions only)
F	First	Treats as a first time balance item (applies to accounts dimensions only)
L	Last	Treats as a last time balance item (applies to accounts dimensions only)

These are the valid **Skip** options per Oracle's Documentation:

TYPE	TYPE DESC	TYPE LONG DESCRIPTION
B	Missing and Zeros	Skips #MISSING data and data that equals zero when calculating the parent value
M	Missing	Skips #MISSING data when calculating the parent value
Z	Zeros	Skips data that equals zero when calculating the parent value

### There's more...

Using the parent-child reference table structure will depend on whether we know that our Measures and Accounts are going to change often. The structure of your fact table will have to change if you decide to use Measure tables. A fact table that has the Measures going down a table vertically, as rows, will allow us to use the Measures column in the fact table to join to the MEASURES table. The following screenshot illustrates how this design will look:



We can easily add accounts or change parent-child associations using this format without having to modify the fact table. On the other hand, if our fact table has Measures horizontally, in columns, then the Measures dimension will have to be built in Essbase Studio or Essbase Integration Services instead. The following screenshot is an example of what a fact table, with Measures as columns, would look like:

SALESFACT	
STATEID	
PRODUCTID	
SCENARIOID	
SUPPLIERID	
TRANSDATE	
SALES	
COGS	
MARKETING	
PAYROLL	
MISC	
OPENINGINVENTORY	
ADDITIONS	

**The Beverage Company (TBC)** sample database's SALES and SALESFACT tables are examples of the two different formats.

## See also

You can find an example of the MEASURES dimension being built in the recipe *Creating hierarchies using a Parent-child reference table* in *Chapter 3*. For an example on how to build the MEASURES dimension using Essbase Studio from the fact table, refer to the recipe *Building a Measures dimension from the fact table* in *Chapter 4*.

## Setting up dimensions with a generation reference

In this recipe, we will build a table in a generation reference format. The SUPPLIER is a geographical dimension. Geographic dimensions are natural hierarchies, which means that the generations are related to each other naturally and there is normally a one-to-many relationship. A generation reference format is common in a relational environment as it can be used to conduct relational reporting as well. The same cannot be said about the parent-child structure.


## Getting ready

To get started, open your SQL Server Management Studio, and add a **TBC** database. Add a **SCHEMA** using a tool such as TOAD, SQL Developer, or Golden, if you are using Oracle.

## How to do it...

1. Run the following scripts to create the SUPPLIER table. We can change the script below to PL/SQL by replacing `int` with `INTEGER` and `varchar()` with `VARCHAR2()`. Following the scripts is a screenshot of the table:

```
--This is the syntax in T-SQL
create table SUPPLIER
(
    SUPPLIERID          int                not null,
    SUPPLIER_ALIAS      varchar(50)        null  ,
    ADDRESS              varchar(25)        null  ,
    CITY                varchar(25)        null  ,
    STATE               varchar(25)        null  ,
    ZIP                 varchar(20)        null  ,
    COUNTRY              varchar(25)        null  ,
    constraint PK_SUPPLIER primary key (SUPPLIERID)
)
go
```

SUPPLIER	
	SUPPLIERID
	SUPPLIER_ALIAS
	ADDRESS
	CITY
	STATE
	ZIP
	COUNTRY

2. Execute the following scripts to add data to the SUPPLIER table:

```
INSERT INTO SUPPLIER
(SUPPLIERID,SUPPLIER_ALIAS,ADDRESS,CITY,STATE,ZIP,COUNTRY)
VALUES (1,'High Tech Drinks','1344 Crossman
Ave','Sunnyvale','California','94675','USA');
```

```
INSERT INTO SUPPLIER
(SUPPLIERID,SUPPLIER_ALIAS,ADDRESS,CITY,STATE,ZIP,COUNTRY)
VALUES (2,'East Coast Beverage','900 Long Ridge
Rd','Stamford','Connecticut','92001','USA');
```

```
INSERT INTO SUPPLIER
(SUPPLIERID,SUPPLIER_ALIAS,ADDRESS,CITY,STATE,ZIP,COUNTRY)
VALUES (3,'Cool Canadian','1250 Boul Rene
Levesque','Montreal','New York','H3B-W4B','Canada');
```

3. Select from the SUPPLIER table to see the results:


```
Select * From SUPPLIER;
```

## How it works...

In step 1, the SUPPLIER table was created and in step 2 the data was populated. A generation in Essbase begins with generation 1 at dimension because the name of the cube in the outline is generation 0. We can tell from the structure of the table that it is clearly set up in **generation reference** as depicted in the following grid:

COLUMN	DESCRIPTION
SUPPLIERID	The PRIMARY KEY and a FOREIGN KEY
COUNTRY	Generation 2
STATE	Generation 3

COLUMN	DESCRIPTION
CITY	Generation 4
ZIPCODE	Generation 5
ADDRESS	Generation 6

 The generation reference will allow us to create ragged hierarchies, but requires the handling of null values by your development tool.

### See also

For more information on how to build the SUPPLIER dimension using Essbase Studio, refer to the recipe *Creating hierarchies using a Generation reference table* in Chapter 3.

## Adding columns for outline formulas

In this recipe, we will add columns to our MEASURES table, so that we can later add a formula to the dimension's members. The importance of this is apparent when you consider that the **Aggregate Storage (ASO)** model does not use the same syntax as the **Block Storage (BSO)** model for their outline formulas. The ASO outline uses **Multidimensional Expressions (MDX)**, which is the standard syntax convention for OLAP applications. We can use our table for both BSO and ASO applications by adding an additional column for the ASO model's formulas.

### Getting ready


To get started, open SQL Server Management Studio, and add a database called TBC. In this recipe, we are using T-SQL, but the PL-SQL equivalent for the examples has been included in the following code snippet. The MEASURES dimension was created in the recipe *Setting up an Account or Measure dimension with parent-child reference* in Chapter 1. We need to complete step 1 of the aforementioned recipe before we continue.

### How to do it...

1. Execute the following script to add a column to the MEASURES table. Following the script is the screenshot of the table after the modification:

```
--This is the syntax in T-SQL
Alter Table MEASURES Add FORMULA_MDX VARCHAR(4000) NULL;
--This is the syntax in PL\SQL
Alter Table MEASURES ADD FORMULA_MDX VARCHAR2(4000) NULL;
```

```
--Delete content of the table to avoid issues with executing this
  exercise
Delete From MEASURES;
```

MEASURES	
	SORTKEY
	MEASURESID
	PARENT
	CHILD
	MEASURES_ALIAS
	CONSOLIDATION
	TWOPASSCALC
	STORAGE
	VARIANCEREPORTING
	TIMEBALANCE
	SKIP
	UDA
	FORMULA
	COMMENT_ESSBASE
	FORMULA_MDX

2. Execute the following script to add a row with the new formula:

```
--This is the syntax for both T-SQL and PL\SQL
INSERT INTO MEASURES (SORTKEY,MEASURESID,PARENT,CHILD,MEASURES_
  ALIAS,CONSOLIDATION,TWOP
    ASSCALC,STORAGE,VARIANCEREPORTING,TIMEBALANCE,SKIP,UDA,FORMULA,
    COMMENT_ESSBASE, FORMULA_MDX) Values(0, 14, 'Ratios', 'Margin
    %', '', '+', 'T', 'X', '', '', '', '', 'Margin % Sales;', '',
    '[Measures].[Sales] / [Measures].[Margin];');

INSERT INTO MEASURES (SORTKEY,MEASURESID,PARENT,CHILD,MEASURES_
  ALIAS,CONSOLIDATION,TWOP
    ASSCALC,STORAGE,VARIANCEREPORTING,TIMEBALANCE,SKIP,UDA,FORMULA,
    COMMENT_ESSBASE, FORMULA_MDX) Values(0, 15, 'Ratios', 'Profit
    %', '', '~', 'T', 'X', '', '', '', '', 'Profit % Sales;', '',
    '[Measures].[Sales] / [Measures].[Profit];');
```

## How it works...

In step 1, the column `FORMULA_MDX` is added to the `MEASURES` table. The script in step 2 adds the new rows with the `FORMULA_MDX` column included. The objective of this recipe is to show you that the syntax is different every time you use a table for both an ASO and BSO set of applications, so you need to have two formula columns. You can see how different the syntax is in the following code snippet, but if you need a more detailed explanation on this, please visit: <http://www.oracle.com/technetwork/middleware/bi-foundation/4395-calc-to-mdx-wp-133362.pdf>. This is Oracle's white paper on *Converting Calc Formulas to MDX in an Essbase Outline*.

FORMULA	FORMULA_MDX
Margin % Sales;	Measures.Sales / Measures.Margin;
Profit % Sales;	Measures.Sales / Measures.Profit;

## Adding the solve order column to tables that have ASO formulas

In this recipe, we will include an additional column to our `MEASURES` table to specify the solve order for the hierarchy. The ASO outline does not have the Two Pass Calc option in its Account dimension; as a result, you will have to specify the solve order by adding an additional column.


## Getting ready

To get started, open SQL Server Management Studio, and add a database called `TBC`. In this recipe, we are using T-SQL, but the PL/SQL equivalent is provided in the examples. The `MEASURES` dimension was created in the recipe *Setting up an Account or Measure dimension with parent-child reference in Chapter 1*. We need to complete step 1 of the aforementioned recipe before we continue.

## How to do it...

1. Execute the following script to add the `FORMULA_MDX` and `SOLVE_ORDER` columns to the `MEASURES` table, if it does not exist:

```
--This is the script in T-SQL
Alter Table MEASURES Add FORMULA_MDX VARCHAR(4000) NULL;
Alter Table MEASURES Add SOLVE_ORDER INT NULL;
--This is the script in PL-SQL
Alter Table MEASURES ADD FORMULA_MDX VARCHAR2(4000) NULL;
Alter Table MEASURES Add SOLVE_ORDER INTEGER NULL;
```

MEASURES	
	SORTKEY
	MEASURESID
	PARENT
	CHILD
	MEASURES_ALIAS
	CONSOLIDATION
	TWOPASSCALC
	STORAGE
	VARIANCEREPORTING
	TIMEBALANCE
	SKIP
	UDA
	FORMULA
	COMMENT_ESSBASE
	FORMULA_MDX
	SOLVE_ORDER

2. Execute the following scripts to add the formula and the solve order values to the MEASURES table:

```
INSERT INTO MEASURES
(SORTKEY,MEASURESID,PARENT,CHILD,MEASURES_ALIAS,
CONSOLIDATION,TWOP ASSCALC,STORAGE,VARIANCEREPORTING,
TIMEBALANCE,SKIP,UDA,FORMULA,COMMENT_ESSBASE,FORMULA_MDX,
SOLVE_ORDER) Values(0, 14, 'Ratios', 'Margin %', '', '+', 'T',
'X', '', '', '', '', 'Margin % Sales;', '', '
[Measures].[Sales] / [Measures].[Margin];', 20);

INSERT INTO MEASURES
(SORTKEY,MEASURESID,PARENT,CHILD,MEASURES_
ALIAS,CONSOLIDATION,TWOP
ASSCALC,STORAGE,VARIANCEREPORTING,TIMEBALANCE,SKIP,UDA,FORMULA,
COMMENT_ESSBASE, FORMULA_MDX, SOLVE_ORDER) Values(0, 15, 'Ratios',
'Profit %', '', '~', 'T', 'X', '', '', '', 'Profit % Sales;',
'', ' [Measures].[Sales] / [Measures].[Profit];', 20);
```

3. Select from the table to see the values that you added:

```
Select * From MEASURES;
```



## How it works...

We started this recipe by adding the `SOLVE_ORDER` column to the `MEASURES` table. We also added two new rows with the `SOLVE_ORDER` populated. The objective of this recipe is to show you that the `SOLVE_ORDER` value has to be higher than its respective components in order for the formula to return the correct values. We should consider the following steps when assigning `SOLVE_ORDER`:

1. Set up in `SOLVE_ORDER` in increments of tens or twenties for clarity and consistency.
2. When the default is not specified, `SOLVE_ORDER` is zero, but it is good practice to always specify the `SOLVE_ORDER` to remove ambiguity and define the calculation's priority.

## Adding and populating the Sort Order Column

In previous releases of Essbase, a developer had the option of building a hierarchy in ascending or descending alphabetical order via a build rule. If you wanted to sort the hierarchies in a different order, then you would go into **Essbase Administrative Services (EAS)**. Then, open the outline, and drag and drop the members in the order that the business wanted or extract the dimension using an Outline Extract utility, sort the hierarchy, and use a build rule to rebuild the dimension. In contrast, when we are using Essbase Studio, in version 11.1.2.1, we are going to have to define the Sort Order in the relational environment. If you have the Oracle's data-governance software **Data Relationship Management (DRM)**, this task will be handled there, but this recipe shows you how to load the Sort Order field with some SQL knowledge and Excel.

## Getting ready

To get started, open SQL Server Management Studio, and add a database called TBC, if you have not already done it. In this recipe, we are using T-SQL and providing the PL/SQL equivalent where the syntax is different. You need to add a `SCHEMA` instead and use a tool like TOAD or Golden, if you are using Oracle. You should also open an Excel workbook.

## How to do it...

1. Execute the following query to create the `YEARS` table. We can change the script below to PL/SQL by replacing `int` with `INTEGER` and `varchar()` with `VARCHAR2()`:

```
--For T-SQL user
Create Table YEARS (
    YEARID      int          NOT NULL,
    YEAR        int          NULL,
```

```

QUARTER      varchar(80)      NULL,
MONTH        varchar(80)      NULL,
MONTH_ALIAS  varchar(80)      NULL,
Constraint PK_YEAR_YEARID Primary Key(YEARID Asc)
);

```

2. Execute the following script to add the **SORT\_ORDER** column:

```

--This is the syntax in T-SQL
Alter Table YEARS Add SORT_ORDER INT NULL;
--This is the syntax in PL/SQL
Alter Table YEARS Add SORT_ORDER INTEGER NULL;

```

3. Open Excel and enter the YEARS dimension's data starting with field **A1**, as follows:

	A	B	C	D	E
1	1	2011	QTR1 11	Jan 2011	January 2011
2	2	2011	QTR1 11	Feb 2011	February 2011
3	3	2011	QTR1 11	Mar 2011	March 2011
4	4	2011	QTR2 11	Apr 2011	April 2011
5	5	2011	QTR2 11	May 2011	May 2011
6	6	2011	QTR2 11	Jun 2011	June 2011
7	7	2011	QTR3 11	Jul 2011	July 2011
8	8	2011	QTR3 11	Aug 2011	August 2011
9	9	2011	QTR3 11	Sep 2011	September 2011
10	10	2011	QTR4 11	Oct 2011	October 2011
11	11	2011	QTR4 11	Nov 2011	November 2011
12	12	2011	QTR4 11	Dec 2011	December 2011

4. Sort the hierarchy manually, if it does not look right in the order specified in the preceding screenshot. Enter the number 1 in cell **F1** and formula =F1+1 in cell **F2**.

	A	B	C	D	E	F
1	1	2011	QTR1 11	Jan 2011	January 2011	1
2	2	2011	QTR1 11	Feb 2011	February 2011	2

- Click on cell **F2**, then click on the box to the right and bottom of the cell, and drag it down to cell **F12**.

	A	B	C	D	E	F
1	1	2011	QTR1 11	Jan 2011	January 2011	1
2	2	2011	QTR1 11	Feb 2011	February 2011	2
3	3	2011	QTR1 11	Mar 2011	March 2011	3
4	4	2011	QTR2 11	Apr 2011	April 2011	4
5	5	2011	QTR2 11	May 2011	May 2011	5
6	6	2011	QTR2 11	Jun 2011	June 2011	6
7	7	2011	QTR3 11	Jul 2011	July 2011	7
8	8	2011	QTR3 11	Aug 2011	August 2011	8
9	9	2011	QTR3 11	Sep 2011	September 2011	9
10	10	2011	QTR4 11	Oct 2011	October 2011	10
11	11	2011	QTR4 11	Nov 2011	November 2011	11
12	12	2011	QTR4 11	Dec 2011	December 2011	12

- Enter the following concatenation string in cell **G1**, select **G1**, and press **CTRL+C**. Select range **G2:G12**, and press **CTRL+V** to paste the concatenation string:  
`= "Insert Into TIME Values (" & A1 & ", " & B1 & ", " & C1 & ", " & D1 & ", " & E1 & ", " & F1 & ");"`
- Copy range **G1:G12**, open up SQL Management Studio, connect to the TBC database, paste the range in the query window, and execute the following queries:

```

Insert Into YEARS Values(1, 2011, 'QTR1 11', 'Jan 2011',
    'January 2011', '1');
Insert Into YEARS Values(2, 2011, 'QTR1 11', 'Feb 2011',
    'February 2011', '2');
Insert Into YEARS Values(3, 2011, 'QTR1 11', 'Mar 2011',
    'March 2011', '3');
Insert Into YEARS Values(4, 2011, 'QTR2 11', 'Apr 2011',
    'April 2011', '4');
Insert Into YEARS Values(5, 2011, 'QTR2 11', 'May 2011',
    'May 2011', '5');
Insert Into YEARS Values(6, 2011, 'QTR2 11', 'Jun 2011',
    'June 2011', '6');Insert Into YEARS Values(7, 2011,
    'QTR3 11', 'Jul 2011', 'July 2011', '7');
Insert Into YEARS Values(8, 2011, 'QTR3 11', 'Aug 2011',
    'August 2011', '8');
Insert Into YEARS Values(9, 2011, 'QTR3 11', 'Sep 2011',
    'September 2011', '9');
Insert Into YEARS Values(10, 2011, 'QTR4 11', 'Oct 2011',
    'October 2011', '10');
    
```

```

Insert Into YEARS Values(11, 2011, 'QTR4 11', 'Nov 2011',
    'November 2011', '11');
Insert Into YEARS Values(12, 2011, 'QTR4 11', 'Dec 2011',
    'December 2011', '12');

```

## How it works...

The following are the steps in this recipe:

1. We added the YEARS table to the TBC database.
2. We added the SORT\_ORDER column to the YEARS table.
3. We added an integer used to sort the members.
4. We also entered the YEARS dimension into an Excel sheet and sorted our YEARS hierarchy.
5. After placing the SORT\_ORDER into column **F1**, we pasted the correct SORT\_ORDER and concatenate Insert statements together with the values in Excel.
6. Finally, we used the Insert statements in the Excel workbook to update the YEARS table using the SQL Management Studio.

The following is what your YEARS hierarchy should look like without the SORT\_ORDER column:

2011
QTR1 11
Feb 2011 Jan 2011 Mar 2011
QTR2 11
Apr 2011 Jun 2011 May 2011
QTR3 11
Aug 2011 Jul 2011 Sep 2011
QTR4 11
Dec 2011 Nov 2011 Oct 2011

Essbase Studio will enter February into the outline before January, May will be after June, August will be before July, and the fourth quarter will be completely out of order. For this reason, it is suggested that you add a `SORT_ORDER` column to all of your dimension tables.

## See also

Refer to the *Using Sort Order on data elements* recipe in *Chapter 2* to learn how to set the sort order for your metadata elements.

## Adding tables for varying attributes

The varying attributes is an attribute dimension that maps to multiple dimensions. The concept of varying attributes in a relational environment is depicted by creating a mapping table. In this recipe, we will build a mapping table that joins the `SALESMAN` table to the `Product` and `Market` tables. We will also see how this format works for a varying attribute.

## Getting ready

To get started, open SQL Server Management Studio, and add a database called `TBC`. In this recipe, we are using T-SQL, but the PL/SQL equivalent is provided in the examples.

## How to do it...

1. Create and populate a `SALESMAN` table with following script. We can change the script below to PL/SQL by replacing `int` with `INTEGER` and `varchar()` with `VARCHAR2()`:

```
--This is the syntax in T-SQL
Create Table SALESMAN(
    SALESMANID          int              NOT NULL,
    SALESMANNAME        varchar (80)     NULL,
    Constraint PK_SALESMAN_SALESMANID Primary Key (SALESMANID)
);
go
--This scripts enter the data into the SALEMAN table
Insert Into SALESMAN Values(1, 'John Smith');
Insert Into SALESMAN Values(2, 'Jose Garcia');
Insert Into SALESMAN Values(3, 'Johnny Blaze');
```

2. Create and populate a `PRODUCTS` table:

```
--This is the syntax in T-SQL
Create Table PRODUCTS(
    PRODUCTID          int              NOT NULL,
    SKU                 varchar(15)     NULL,
```

```

        SKU_ALIAS    varchar(25) NULL,
        Constraint PK_PRODUCTS_PRODUCTID Primary Key (PRODUCTID)
    );
--Insert data into PRODUCTS table
Insert Into PRODUCTS Values(1, '100-10', 'Cola');
Insert Into PRODUCTS Values(2, '100-20', 'Diet Cola');
Insert Into PRODUCTS Values(3, '100-30', 'Caffeine Free Cola');

```

**3. Create and populate a MARKETS table:**

```

--This is the syntax in T-SQL
Create Table MARKETS(
    STATEID        int            NOT NULL,
    STATE          varchar(25)    NULL,
    Constraint PK_MARKETS_STATEID Primary Key (STATEID)
);
--Insert data into MARKETS table
Insert Into MARKETS Values(1, 'New York');
Insert Into MARKETS Values(2, 'Massachusetts');
Insert Into MARKETS Values(3, 'Florida');

```

**4. Create and populate a SALESMANMAP table:**

```

--This is the syntax in T-SQL
Create Table SALESMANMAP(
    SALESMANID    int NOT NULL,
    STATEID       int NOT NULL,
    PRODUCTID     int NOT NULL,
    Constraint PK_SALESMANMAP Primary Key (STATEID, PRODUCTID)
);

--Insert data into the SALESMANMAP table
Insert Into SALESMANMAP Values(1,1,2);
Insert Into SALESMANMAP Values(1,2,3);
Insert Into SALESMANMAP Values(2,1,1);
Insert Into SALESMANMAP Values(2,2,2);
Insert Into SALESMANMAP Values(3,3,1);
Insert Into SALESMANMAP Values(3,3,3);

```

**5. Execute the following scripts to join the tables:**

```

Alter Table SALESMANMAP Add Constraint FK_SALESMANMAP_PRODUCTID
Foreign Key(PRODUCTID)
References PRODUCTS (PRODUCTID);

Alter Table SALESMANMAP Add Constraint FK_SALESMANMAP_SALESMANID
Foreign Key(SALESMANID)
References SALESMAN (SALESMANID);

```

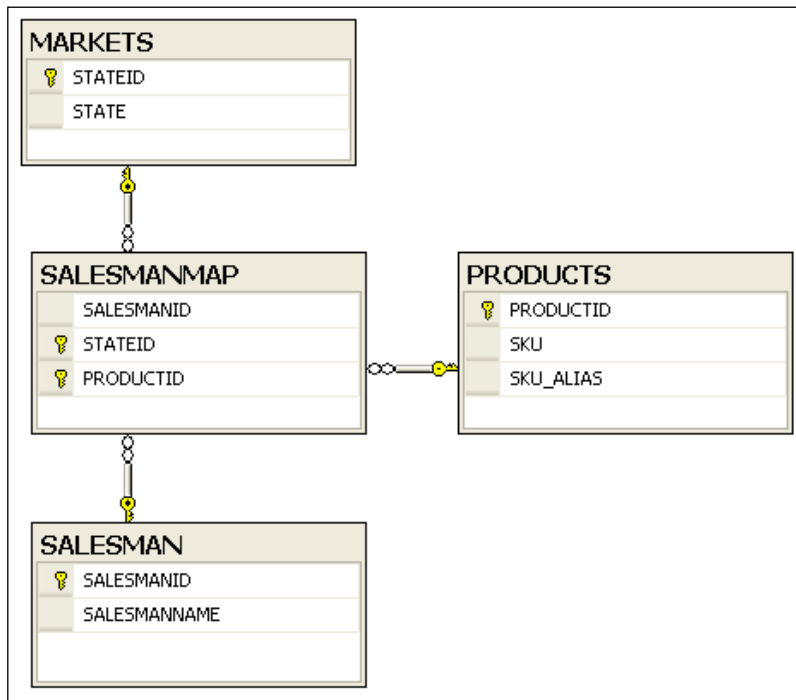
```
Alter Table SALESMANMAP Add Constraint FK_SALESMANMAP_STATEID  
Foreign Key (STATEID)  
References MARKETS (STATEID);
```

6. Execute the following query to see the relationship between the Market, Product, and Salesman:

```
Select  
T4.SALESMANID, T2.SALESMANNAME, T4.STATEID,  
T3.STATE, T4.PRODUCTID, T1.SKU_ALIAS  
From PRODUCTS T1, SALESMAN T2, MARKETS T3, SALESMANMAP T4  
Where T1.PRODUCTID = T4.PRODUCTID and  
T2.SALESMANID = T4.SALESMANID and  
T3.STATEID = T4.STATEID and  
T2.SALESMANNAME = 'John Smith'
```

### How it works...

The relationships between the tables created and populated in steps 1 through 4 are shown in the following diagram:



The **SALESMANMAP** table joins the other three tables of which **SALESMAN** is the varying attribute. Salesman varies depending on the state and product you want to report on. We can retrieve the relationship between tables using the query shown in step 5. The results for this query are displayed in the following screenshot:

	SALESMANID	SALESMANNAME	STATE...	STATE	PRODUCTID	SKU_ALIAS
1	1	John Smith	1	New York	2	Diet Cola
2	1	John Smith	2	Massachusetts	3	Caffeine Free Cola

The important thing to note about this varying attribute example is that it is possible for two Salesmen to conduct business in the same Market, but it is not possible for the same two Salesmen to sell the same Product in the same Market. This logic is maintained by the constraints placed on the **SALESMANMAP** mapping table.

## See also

Refer to the *Adding tables to a Minischema* recipe in *Chapter 2* to add the varying attribute example to the TBC database. Refer to the *Setting Essbase Properties* recipe in *Chapter 3* to learn how to set the varying attributes' properties.

## Determining hierarchies in relational tables

In this recipe, we will determine hierarchies in relational models. This recipe will also go over some of the main attribute dimension types. Attribute dimensions are dynamic dimensions that allow users to report on their data without increasing the foot print of the database. Attributes work in a similar way to an alternate hierarchy, but unlike an alternate hierarchy you can use an attribute dimension to conduct cross tab reporting on a different axis than your base dimension.

## Getting ready

To get started, open SQL Server Management Studio, and add a database TBC, or if you are using Oracle you can add schema TBC and use TOAD or Golden to complete the recipe.

## How to do it...

1. Execute the following script to add the Product Table. We can change the script below to PL/SQL by replacing `int` with `INTEGER` and `varchar()` with `VARCHAR2()`:

```
--Create Product Table T-SQL
create table PRODUCT
(
```