

Oracle Express Server™ Database Administration Guide

Release 6.0

Part No. A47232-1

ORACLE®

Copyright © 1996, Oracle Corporation. All rights reserved.

This software contains proprietary information of Oracle Corporation; it is provided under a license agreement containing restrictions on use and disclosure and is also protected by copyright law. Reverse engineering of the software is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. Oracle Corporation does not warrant that this document is error-free.

RESTRICTED RIGHTS LEGEND

Programs delivered subject to the DOD FAR Supplement are “commercial computer software” and use, duplication and disclosure of the Programs shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are “restricted computer software” and use, duplication and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-14, Rights in Data -- General, including Alternate III (June 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be licensee's responsibility to take all appropriate fail-safe, back up, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and Oracle disclaims liability for any damages caused by such use of the Programs.

Oracle and Express are registered trademarks of Oracle Corporation, Redwood City, California. Oracle Express Server and Oracle Sales Analyzer are trademarks of Oracle Corporation, Redwood City, California.

All other company or product names mentioned are used for identification purposes only and may be trademarks of their respective owners.

Contents

	About This Manual	vii
	Administrator Documentation	vii
	Conventions	x
Chapter 1	Introduction	1
	What Is Express Server?	1
	What Is a Database?	2
	Relational vs. Multidimensional Databases	12
Chapter 2	Connecting to Express	17
	Client Application Requirements	17
	Overview of Connecting	18
	Configuring for Connection to Express Server	19
Chapter 3	Getting Started	31
	What Is Administrator?	32
	Starting and Exiting Administrator	36
	Understanding the Main Window	37
Chapter 4	Creating a Database	49
	The Database Dictionary	49
	Creating a New Database	50
	Upgrading Express 5.x Databases to Express 6.0	70

Chapter 5	Structuring Data	75
	Working with Databases 75	
	Working with Dimensions 85	
	Working with Measures 89	
	Executing Express Commands 100	
Chapter 6	Importing and Exporting Data	103
	Importing Data 103	
	Exporting Data to an EIF File 108	
Chapter 7	Manipulating Data	111
	Creating Valuesets 111	
	Creating Programs 113	
	Copying, Renaming, and Deleting Objects 117	
	Setting Properties 119	
	Searching for Objects 121	
Chapter 8	Configuring Databases for a Client Application	123
	The Configuration Check List 124	
	Installing Databases, Dimensions, and Measures 125	
	Hiding Dimensions and Measures 126	
	Uninstalling Databases, Dimensions, and Measures 127	
	Editing Dimension Hierarchies 128	
	Defining a Dimension's Selections 138	
	Specifying Default Relations 139	
	Defining System Shutdown Programs 142	
	A Simple Case Study 143	
Chapter 9	Managing a Database	149
	Using Express Commands 149	
	Improving Database Efficiency 153	
	Making Calculations Efficiently 154	
	Planning for a Smaller Database 156	
	Keeping Your Database Efficient 188	
	Saving a Database in Two or More Files 192	
	Controlling Access to Your Database 194	

Chapter 10	Configuring for SQL and XCA Connections	207
	Setting Up the System for SQL Connections	207
	Setting Up a Personal Express XCA Connection to Express Server	216
Appendix A	Administrator Installation	227
	Hardware and Software Requirements	227
	Installing Administrator	228
	Starting and Exiting Administrator	233
Appendix B	Using Conjoint Dimensions	237
	Defining and Maintaining Conjoint Dimensions	237
	Converting Conjoint Dimensions to Composites	240
	Converting Composites to Conjoint Dimensions	241
	Index	243

About This Manual

This manual describes Oracle Express Administrator™ (hereinafter referred to as “Administrator”) for client applications.

You use Administrator to create Express® databases and to customize them to optimize the way client applications display data. The configuration procedures described in this manual are optional. However, many of the features that make client applications powerful and easy to use are available only if you have configured your database.

This manual is intended for a database administrator (DBA) who manages the databases shared by a community of users. These users might create projects, queries, or briefings and distribute them to other client application users in the community. Since data configuration information must be consistent among all the users, it is important that a database administrator coordinate configuration activities for all the users in a community.

The reader of this manual should also be familiar with Oracle®’s prebuilt client application products, which are described in Chapter 2. Since the effects of the configuration procedures will be seen in tables, graphs, and spreadsheets, as well as in the Selector and Database Browser in these products, the DBA doing the configuration should understand how these components display data for the user.

Administrator Documentation

This manual is part of a comprehensive documentation set. The following sections describe the documentation that is provided for Administrator and for other products with which this product runs.

This Manual

The *Database Administration Guide* provides conceptual information and some procedural information for DBAs using Administrator. For complete procedural information and a glossary of terms, consult the Administrator Help system.

This manual contains the following chapters:

- **Chapter 1** introduces you to Oracle Express Server™ (hereinafter referred to as “Express Server”) and the objects that make up an Express Server database.
- **Chapter 2** describes how to connect to Express Server 6.0 for NT.
- **Chapter 3** provides an overview of Administrator and describes how to get started.
- **Chapter 4** describes how data is organized in the database dictionary and how to create a new database.
- **Chapter 5** describes how to structure your data by explaining how to work with databases, dimensions, and measures.
- **Chapter 6** describes how to import data into Express databases and how to export data from them.
- **Chapter 7** describes how to manipulate your data by working with objects such as programs and models and by copying, renaming, deleting, and searching for objects.
- **Chapter 8** describes how to configure databases by providing a configuration check list and detailing the steps in the process.
- **Chapter 9** describes methods you can use to improve the efficiency of your database.
- **Chapter 10** describes how to configure your system to use SQL data and Express data.
- **Appendix A** describes gives detailed instructions for installing Administrator.
- **Appendix B** describes how to use conjoint dimensions.

Many of the examples in this manual are taken from the XADEMO database, which can be installed with Express Server.

Other Documents for Administrator

- Oracle Express Administrator Help system — Provides procedural and reference information for using Administrator, both to create and maintain databases and to optimize existing Express databases for use with client applications.

Related Documentation

This manual is part of a comprehensive set of documentation that also includes:

- *Installation and Configuration Guide* — Explains how to install, configure, and manage Express Server as a service on a Windows NT server or workstation.
- *Express Language Reference Manual* — Explains the syntax and usage of the individual commands, functions, options, and programs that make up the Express language.
- *Express Language Programming Guide* — Explains the Express programming environment and illustrates programming strategies for accessing and analyzing data.
- *API Guide* — Explains how to use SNAPI to develop client applications that interact with Express Server.
- *Express Excel Add-In User's Guide* — Explains how to use the Excel Add-In to develop client applications in Excel that interact with Express Server.

In addition to these manuals, the following Help systems are available with Express 6.0:

- *Express Installation Help* — Provides online Help for installation.
- *Express Service Manager Help* — Provides online Help for managing services on your Windows NT system.

- *Express Configuration Manager Help* — Provides online Help for selecting Express modules and configuring runtime settings.
- *Express Language Help* — Provides online Help for the syntax and usage of the commands, functions, options, and programs that make up the Express language.
- *Web Developer's Toolkit Help* — Provides online Help for the XWHTML and XWDEVKIT databases, which contain programs for generating HTML tags and tables of Express data in HTML documents.
- *Excel Add-In Help* — Provides online Help for the Excel Add-In.
- *Express Session Manager Help* — Provides online Help for managing individual Express sessions.

Conventions

This manual uses a set of terms, symbols, and typographic conventions to categorize specific information. Familiarity with these conventions will help you use this manual more effectively.

The following icons are used in this manual.

Icon	Meaning
	A procedure that you should follow to complete a task
	References to other manuals or other sections of this manual
	References to the Help system

You will find the following text conventions in this manual.

Convention	Usage
UPPERCASE TEXT	Indicates directory names, file names, operating system commands, Express commands and objects, and acronyms.

<code>Courier text</code>	Indicates examples and anything that you must type exactly as it appears. For example, if you are asked to type <code>show eversion</code> , you would type all the characters shown in Courier font exactly as they are printed.
boldface text	Indicates menu items, command buttons, options, and field names.
<i>italic text</i>	Indicates variables, including variable text. Variable text is used when dialog boxes or their components are unlabeled or have labels that change dynamically based on their current context. The wording of variable text does not exactly match what you see on your screen. Italic type is also used for emphasis and for titles of documents.

Mouse Usage

This manual includes procedures in which you use both mouse buttons. Always use the left mouse button unless you are specifically instructed to use the right mouse button. The term “left mouse button” refers to the dominant button. If you have reconfigured your mouse to reverse the functions of the left and right buttons, you will need to use the reverse button when following the procedures in this manual.



For definitions of basic mouse terms, refer to the documentation for Microsoft Windows.

Keyboard Formats

All key names are shown using small capital letters. For example, the Control key is shown as CTRL. The UP ARROW, DOWN ARROW, LEFT ARROW, and RIGHT ARROW keys are collectively referred to as the arrow keys.

Key combinations and key sequences appear in the following formats.

Format	Meaning
KEY1+KEY2	A plus sign (+) between key names means to press and hold down the first key while pressing the second key. For example, “press ALT+TAB” means to press and hold down the ALT key while pressing the TAB key.
KEY1, KEY2	A comma (,) between key names means to press and release the keys one after the other. For example, “press ALT, F, O” means to press and release the ALT key, press and release the F key, then press and release the O key.

Chapter 1

Introduction

This chapter introduces you to Oracle Express Server™ (hereinafter referred to as “Express Server”), by describing some of its capabilities and explaining the objects that make up an Express Server database.

Once you understand the data model behind an Express Server database, you will use Oracle Express Administrator™ (hereinafter referred to as “Administrator”), to create and manage databases on Express Server.

This chapter describes:

- Express Server
- An Express Server database
- A comparison of SQL (Structured Query Language) concepts and terms with Express concepts and terms
- Summary

What Is Express Server?

Express Server is a multidimensional database system for analyzing, tracking, and reporting information about your business. It has a complete programming language for developing applications tailored to your company’s needs.

Express Server is a versatile information tool. With its ability to create relationships among your data, it can create a model of your entire business environment.

Express Server has sophisticated data analysis and forecasting tools that you can use for business applications such as:

- Tracking market share
- Analyzing pricing models
- Analyzing promotion and advertising effectiveness
- Long-range planning and budgeting

What Is a Database?

An Express Server *database* is a collection of data that is organized so that you can look at it from different perspectives.

An Express Server database is made up of objects and data. *Objects* are definitions in the database dictionary, which is Express Server's index to the database. They are places to store information, such as data, program code, and dimension values. Express Server has several kinds of objects: dimensions, relations, variables, formulas, programs, models, composites, valuesets, and worksheets.

The most important objects to understand are variables and dimensions:

- Variables are the objects in which you store data.
- Dimensions provide a way to organize that data.

Other Express® objects give you a variety of powerful ways to work with your data.

To get the most out of an Express database, you should first understand its objects and how you can use them to your advantage. The more you understand about an Express database before you create one, the easier your job will be.

When you create an Express database, you must create your dimensions *before* you create your variables. In other words, your job is to determine how to organize data before deciding how to store it. Therefore, the first object you should understand is the dimension.

The following sections describe the objects you can create in an Express database. For information about creating Express database objects, see Chapter 4 and the Administrator Help system.

What Are Dimensions?

Dimensions are the foundation of your database. A *dimension* is a list of values that provides an index to your data. Dimension values let you identify your data and select subsets of data with which to work.

You define dimensions for your database based on how your data is organized and the level of detail you are interested in maintaining.

Table 1.1 lists several ways you can organize your data. One set of data can be organized in several different ways.

Table 1.1 How Data Can Be Organized

Organizing Factor	Example Dimensions
Time	MONTH, QUARTER, YEAR
Geography	COUNTRY, REGION, CITY
Product	PRODUCT, BRAND, CATEGORY
Organization	DIVISION, COMPANY
Financial	ACCOUNT

For example, you might want to study sales data by location, product, and time period. This means you need at least three dimensions. If you do not care about separating data by location, you can total all locations and define sales to be dimensioned by product and time period only.

You also need dimensions for each level of detail in your data. If your company is divided into sales districts and each district handles several store accounts, you need to decide whether you want sales figures for every store or only for each district. If you need store data, you can define a STORE dimension; if you always look at each district as a whole, all you need is a DISTRICT dimension. Or, you can have both a STORE and a DISTRICT dimension. Then, by creating a relation between STORE and DISTRICT, you can organize data by store and view aggregates of data by

district. Relations are described later in this chapter. Although your Express Server database can be as detailed as you want, the use of disk space is most efficient if you include only as much detail as you will need to analyze your data.

Express Server's strength is in processing multidimensional data. You are probably familiar with two-dimensional data from spreadsheets, pictured as a matrix of rows and columns. Figure 1.1 compares a two-dimensional spreadsheet with a multidimensional array. Express Server's dimension values provide descriptive labels for the data, which is more useful than the cell numbers of a typical spreadsheet.

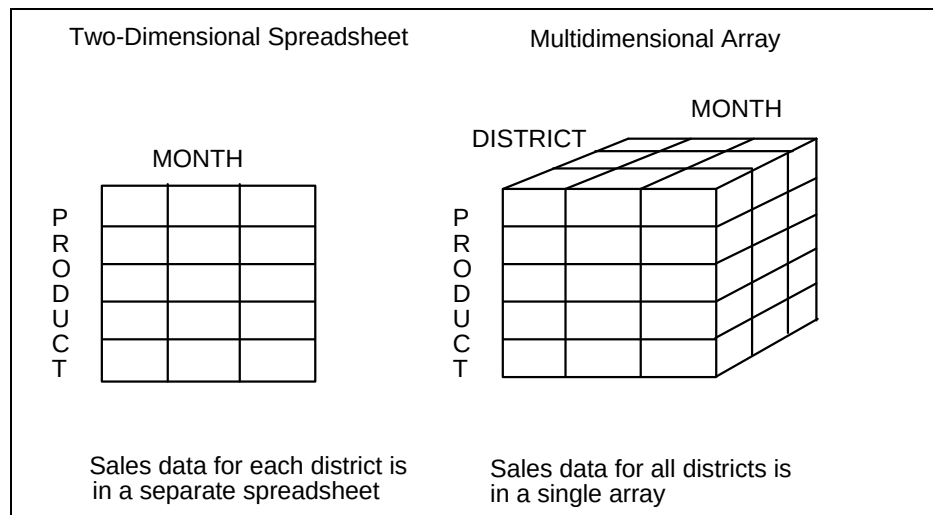


Figure 1.1 A two-dimensional spreadsheet contrasted with an Express multidimensional array.

You can think of three-dimensional data as a cube or as a multilayer spreadsheet. In Express Server, each additional layer is called a *slice* of data. No matter how you define your data, you can view it sliced by any of its dimensions, as illustrated in Figure 1.2.

Note that although the examples and illustrations show data organized by three dimensions, you are not limited to three dimensions. For example, a variable can be dimensioned by a maximum of 32 dimensions.

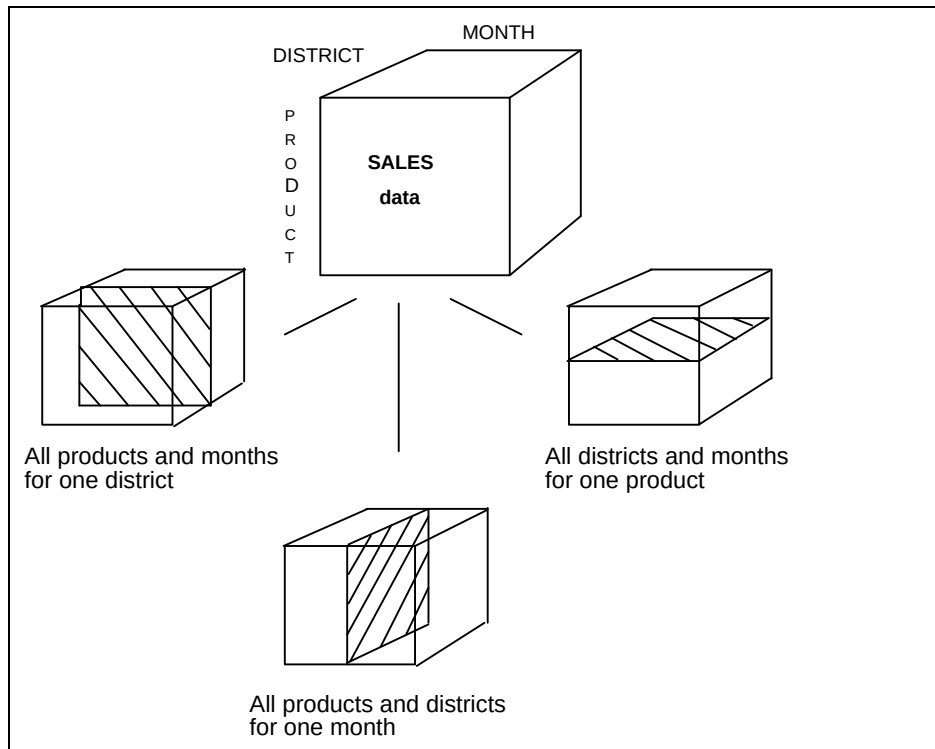


Figure 1.2 Different users can take different views of the same data.

Express Server allows you to define three basic types of dimensions:

- **Time** — Values represent time periods. Express Server provides special features for ease in handling and maintaining the values.
- **Text** — Values are brief descriptions of the entities they represent.
- **Integer** — Values are identified only by their numeric positions.

Table 1.2 summarizes the data types available for each type of dimension.

Table 1.2 Data Types of Dimensions

Dimension Type	Data Type	Used For
Time	DAY	One-day periods.
	WEEK	Periods of 1 to 52 weeks.
	MONTH	Periods of 1, 2, 3, 4, or 6 months.
	QUARTER	Three-month periods.
	YEAR	Twelve-month periods.
Text	ID	Dimension values up to eight characters in length. Express Server processes ID dimensions very efficiently. If your dimension will have a large number of values, you should define it with a data type of ID and use values of eight characters or less. ID dimension values can contain blanks or lowercase letters.
	TEXT	Dimension values of more than eight characters, or values containing blanks or lowercase letters.
Integer	INTEGER	Lists that do not require text labels as dimension values.

Text and INTEGER Dimensions

For dimensions that do not hold time values, text dimensions (with a data type of TEXT or ID) are generally more useful than INTEGER dimensions. The values of text dimensions give a clear description of the data and provide ready-made labels for client applications to use. However, if you cannot find logical unique names for the dimension's values, you can use an INTEGER dimension.

For example, it is easy to create unique names for products and districts and use those names as dimension values. However, an employee dimension using last names would run into trouble if two employees had the same name. Instead, you could have an INTEGER EMPLOYEE_INDEX dimension. An INTEGER dimension is a sequence of numbers — one integer for each dimension value — that always begins with “1” (for example, “1, 2, 3, ... *n*”). Each EMPLOYEE_INDEX dimension value would serve as a unique identifying number for each employee. The employees' last names and other information would be stored in variables

dimensioned by `EMPLOYEE_INDEX`. Another alternative would be to have a text dimension with the employees' social security numbers as dimension values.

What Are Variables?

Each array of data is called a *variable*. A database might contain the `SALES` variable shown above as well as other variables, such as `EXPENSE` and `PRICE`, as illustrated in Figure 1.3.

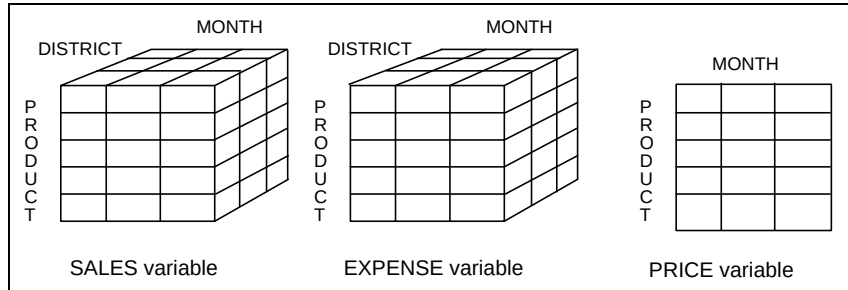


Figure 1.3 Examples of variables in a database.

How Variables Share Dimensions

If two variables use the same dimensions, such as `MONTH` and `PRODUCT`, these dimensions are not stored in the database twice. Instead, the two variables *share* the dimensions. Shared dimensions provide an important link between variables that have some or all of the same dimensions. For example:

- If you look at `SALES` and `EXPENSE` data, notice the same months, products, and districts are used for both variables.
- If you add, delete, or rename products, the products are added, deleted, or renamed for all variables that have the `PRODUCT` dimension.
- If your users select products to use in their client application, the same products are selected for all variables that have the `PRODUCT` dimension.

By sharing dimensions, Express Server ensures the *referential integrity* of your database. Whatever changes you make to the dimension values in your database, those changes are automatically reflected throughout your data. You can always depend on the consistency of your data throughout any reports, graphs, or analyses your users' client applications may use.

Using Dimension Values

Dimension values not only identify your data, but also provide an easy way to target the data you need for a particular purpose. When your users display, graph, or analyze your data, they *select* the values to work with. Figure 1.4 illustrates some selections a user might make while working with SALES data.

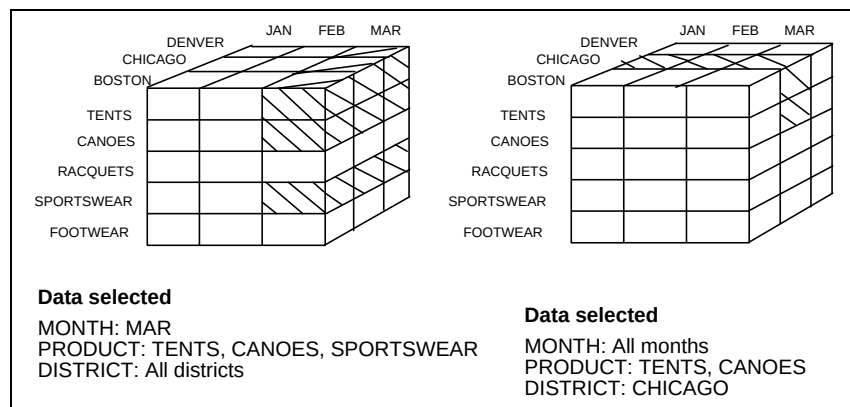


Figure 1.4 Using dimension values to select the data.

Selecting dimension values has no permanent effect on your database. Every dimension has all its values available whenever your users connect to your database, and they can change the selection at any time as they work with your database.

What Are Relations?

A *relation* is a correspondence or association between the values of two dimensions. For example, each product is made by a division of the company, and a relation `DIVISION_PRODUCT`, as shown in Figure 1.5, associates each product with the division that makes it.

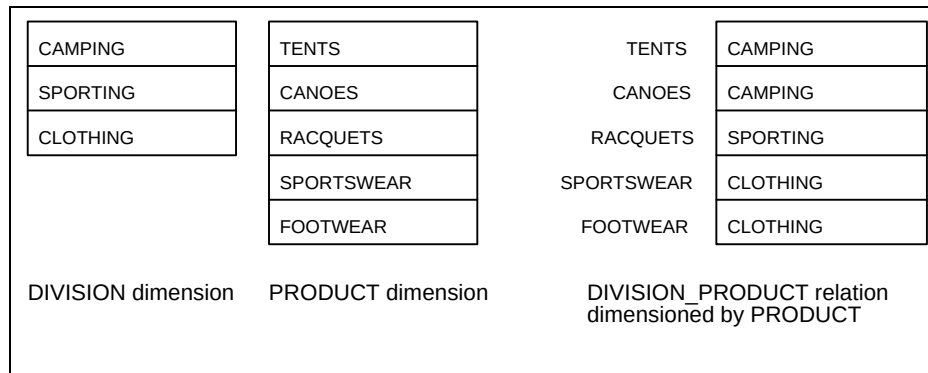


Figure 1.5 Example of a relation between two dimensions.

A relation is usually dimensioned by the dimension with the larger number of values, which is called the *less aggregate* dimension. The value of the relation is a dimension value of the other dimension, or *related dimension*.

The related dimension is usually the dimension with fewer values, which is called the *more aggregate* dimension. The relation is dimensioned by the less aggregate dimension, and every value is explicitly connected to the related dimension.

For example, you can define a relation between a `CITY` dimension and a `REGION` dimension. The dimension of regions has fewer values than the dimension of cities. To assign every city to a region, you need a relation dimensioned by `CITY`, with `REGION` as the related dimension. Every city has a cell where you can store a region. Figure 1.6 illustrates this relation.

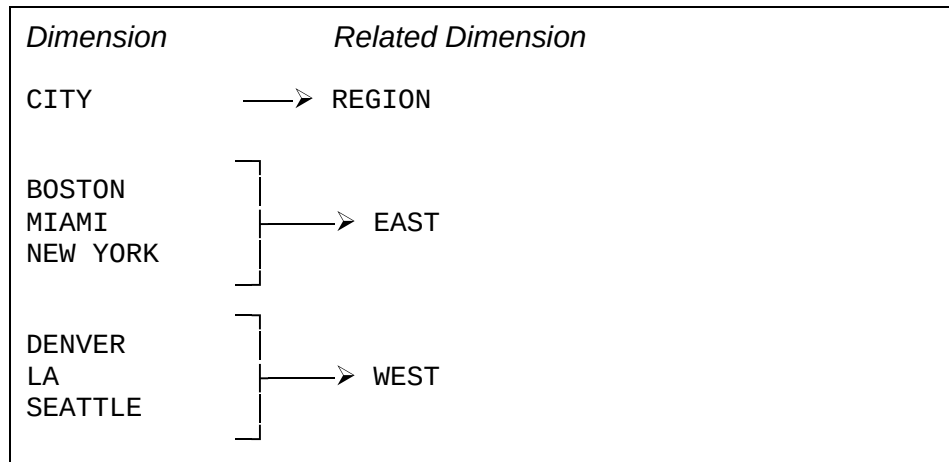


Figure 1.6 Relation between cities and regions.

Express Server automatically creates relations between the time dimensions in your database. These relations are called *implicit* relations because they are not actually defined as objects in your database. However, you can use them in the same way that you use the relations that you explicitly define yourself.

For example, if you define a MONTH dimension, a QUARTER dimension, and a YEAR dimension, Express Server creates implicit relations between MONTH and QUARTER, between MONTH and YEAR, and between QUARTER and YEAR.

You can have a relation that is dimensioned by the dimension with fewer values. In this case, not every value of the other dimension is related. For example, you could create a relationship, named CAPITAL_OF_STATE, between states and their capital cities. The relation is dimensioned by STATE and the related dimension is CITY. Only the capital cities are assigned to a state.

You can have more than one relation between two dimensions. This means that in addition to the relation between the capital cities and their states, which is dimensioned by STATE, you can have another relation, dimensioned by CITY, that assigns every city to its state.

You can use relations to organize dimension values into hierarchical groups. For example, you can use relations to group products into categories and categories into divisions.

When you name a relation, you need a consistent naming convention to make the relationship clear. The name of the related dimension is connected to the dimension of the relation.

You can also have two-dimensional relationships. For example, you might want different groupings of the same dimension values for different analyses. If you group districts into regions differently, for example, depending on the product category, you might define a relation between region and a combination of district and category. If you hire temporary workers and each month they work in a different division, you can have a relation between the division and the combination of employee and month.

Other Express Server Objects

Formulas are objects that define an expression. Formulas calculate temporary data you do not need to store permanently in your database. For example, if sales are always units sold multiplied by price, you can define a formula to calculate sales and avoid storing redundant data in the database. You can analyze the value of sales just as you would any variable; however, every time you refer to sales, the value is recalculated.

If you need to use a very complex calculation, you might want to define it in a program rather than as a formula. If you define a program that uses a RETURN command to return a value, you can use the program as a function wherever you would use an expression or a formula. (An *expression* is one or more data values, which you use as an argument in commands or functions. See the Glossary in the Express Language Help for a detailed definition of “expression.”)

Formulas, variables, and relations are collectively referred to as *measures*. As you use this manual, you will find that many database tasks involve working with measures, because these are the objects your users will be selecting from their client applications.

Composites are objects that store sparse data in a compact form, which saves disk space. A composite is an artificial dimension that combines the

dimension values of two dimensions along which data is sparse. A composite is automatically created by Express when you define a sparse variable.

Programs let you store sequences of commands that manipulate your data. Some commands, which can only be used in programs, enable you to loop over sets of data.

Models let you store and solve a set of interrelated equations. The equations can be based on the values of variables or dimensions. For example, you can define a financial model with equations based on the values of a line item dimension. The equations in the model can use input data for some line items, such as revenue and expenses, to calculate other line items, such as net income and return on sales.

Valuesets let you store a list of dimension values for a particular dimension. The values in a valueset can be saved across Express Server sessions. You can limit a dimension to the values stored in a valueset for that dimension.

Worksheets let you import text data from external files and transfer it to variables and dimensions in your database.

For detailed information about creating and using variables, relations, and formulas, see Chapter 5. For more information about valuesets, programs, and models, see Chapter 7 in this manual, and the *Express Language Programming Guide*. See Chapter 9 for details about composites, sparse data, and sparse variables. See the LIMIT command entry in the Express Language Help for information on valuesets. See the DEFINE WORKSHEET command in the Express Language Help for more information about worksheets.

Relational vs. Multidimensional Databases

Now that you have looked at the model for an Express Server database, it may be helpful to discuss a relational database in comparison to a multidimensional database.

Many businesses use relational databases to store their corporate data. Relational databases offer an efficient method of recording and storing vast amounts of information using an interface created in Structured Query Language (or SQL, usually pronounced “sequel”). SQL is the ANSI-standard programming language for relational databases.

Relational databases are best suited to recording individual transactions rather than, for example, comparing groups of transactions over a period of time. A database designed primarily for online transaction processing (OLTP) does not lend itself easily to the analysis of that information. To support analytical processing, other facilities are needed, such as multidimensional databases and online analytical processing (OLAP) tools.

Express Server databases are designed specifically to support analytical processing and are able to use data from relational databases. Refer to Chapter 10 to learn how to set up an Express-based system for connecting to relational databases to perform dynamic SQL reach-through.

Differences in Database Models

A relational database stores information in tables organized by rows and columns. Each table contains a column or a combination of columns whose values uniquely identify each row. These unique values are called *primary keys* and help ensure the integrity of the data. When the values in a column match the values of another table’s primary key, that column is called a *foreign key*. Relationships between tables are established through primary and foreign keys. You can select columns of data from different tables and view them together as long as the tables are related in this way.

In contrast, Express Server stores information in multidimensional variables. A variable contains information that quantifies a particular aspect of your business. For example, you might record sales data in dollars (a Sales variable) and units (a Units variable). All of the data in a variable represent the same unit of measurement. Your business might have several ways of measuring its transactions — through dollars, units, percentages, and so on — and each one will be stored in its own variable.

It might be helpful to look at SQL concepts and terms in comparison to the equivalent concepts and terms used by Express Server, as shown in the following table.

Note that client application users can specify how they want to view the data from your database by using the Selector, which is a feature of prebuilt client applications.

SQL	Express
Database	Database
Table	N/A
View	Formula
Column	Variable (Array) or dimension
Row	Variable limited by n-1 dimensions
Primary key	Dimension
Foreign key	N/A (dimension only exists once)
Declarative referential integrity	Implied by singular definition of dimensions
Indices	N/A
System catalog	EXPRESS.DB
JOIN clause	Relation
WHERE clause	LIMIT command
GROUP BY clause	GROUP keyword (in commands such as REPORT)
ORDER BY clause	SORT command
GRANT	PERMIT command
Stored procedures, scripts, stored SQL	Programs, user-defined functions, and INFILE command

Control of flow language (Transact-SQL, PL/SQL, ...)	Express programming language
Aggregations (Sum, Avg, Count, Min, Max)	Express functions and formulas using dimensions
SELECT	REPORT command
INSERT	MAINTAIN ADD command
DELETE	DELETE command, MAINTAIN DELETE command
UPDATE	SET command

Chapter 2

Connecting to Express

This chapter explains how to set up Administrator and client applications for connection to Express Server 6.0, using a facility called Remote SNAPI.

This chapter describes the following:

- Client application requirements for connecting to Express Server
- Overview of connecting to Express Server
- Configuring for connection to Express Server
- Summary



If you need to set up Administrator to access SQL data, see Chapter 10.

Client Application Requirements

Oracle® client applications include the Excel Add-In for Express (hereinafter referred to as “the Excel Add-In”), Oracle Express Analyzer™, and Oracle Express Objects. Express Server 6.0 includes the Excel Add-In as a client application. You may purchase Oracle Express Analyzer and Oracle Express Objects separately. (Note that an evaluation copy of Oracle Express Analyzer is included with Express Server 6.0.)

In this manual, Oracle Express Analyzer and Oracle Express Objects are collectively referred to as “the Express Client products.”

The Excel Add-In and the other Express Client products have the same requirements for connecting to Express Server as Administrator.

Be aware that any additions or changes you make to the changes in the XADMIN.INI file must also be made to the .INI file for every client application that wants to connect to Express Server. In other words, the .INI file settings for any client application should be identical to the XADMIN.INI file settings for Administrator.

Overview of Connecting

Much of the power of Administrator and the client applications derives from their ability to understand and access Express data. The Express data resides on the same remote system on which Express Server runs, and Administrator and the client applications communicate directly with Express Server using Remote SNAPI (Structured N-dimensional API). To run applications, users connect to the Express Server machine.

You can use menu items to connect and disconnect Administrator and the client applications to Express while they are running. Alternatively, you can configure Administrator or a client application to display a dialog box, which prompts you to select a server machine, at startup by editing its .INI file, as described later in this chapter.

When Administrator is not connected to Express, the File menu in Administrator contains the **Connect To Express** option. If you choose this menu option, the Connect dialog box is displayed so you can specify a connection to Express. After you specify the appropriate settings in the dialog box and choose **OK**, a connection to Express is established.

When Administrator is connected to Express, the File menu contains the **Disconnect From Express** item. If you choose this menu item, all databases connected through Administrator are detached.

If your users' client application is the Excel Add-In, be aware that the Excel Add-In cannot disconnect from Express. The Excel Add-In connects on an attempt to attach a database. For information, see the *Excel Add-In Installation and User's Guide*.

Configuring for Connection to Express Server

Configuring Administrator and the client applications to connect to Express Server involves three steps, which are described in the following sections.

The following list presents the order in which you should perform the steps:

1. Understand the requirements for connecting to Express Server.
2. Move database files to the Express Server machine.
3. Set .INI file options.

Understanding the Requirements for Connecting to Express Server

In order to connect Administrator and the client applications to Express Server, you must understand the requirements involved in connecting. The following list describes the system requirements for connecting Administrator and the client applications to Express Server.

- **Express Server** — Express Server 6.0 (or later) must be installed on the remote NT system where the Express data will reside. Your system administrator will probably install Express Server.
- **Remote SNAPI** — You must install Remote SNAPI on the system that contains Administrator or a client application.
- **RPC runtime support** — Remote SNAPI uses RPCs (Remote Procedure Calls) to communicate with Express Server. You must install RPC runtime support on the system on which Administrator or a client application is running. Administrator, the Excel Add-In, and the other Express Client products support the freely distributable Microsoft RPC runtime DLLs (MS-RPC).
- **TCP/IP support** — You must install TCP/IP support on the system on which Administrator or a client applications is running. TCP/IP is the network protocol used by Remote SNAPI to communicate with Express Server. TCP/IP implementations are widely available.

Before following the instructions in this chapter, ensure that the products listed above are properly installed on your systems. For information about installing, configuring, and using Remote SNAPI, see the *Oracle Express Server API Guide*.

Database Files on the Express Server Machine

To run a client application or Administrator with Express Server, four Express database files used by these products must be present on the Express Server machine. These application database files are: XPDB.DB, XPDDCODE.DB, XPDDDATA.DB, and XPADMIN.DB.

The Express Server CD includes the application database files. When your system administrator installs Express Server, these application database files are automatically installed and put in a subdirectory on the Express Server machine.

In addition, two demonstration databases (DEMO.DB and XADEMO.DB) are automatically installed on the Express Server machine.

All of these database files are installed with read/write permissions. Your system administrator can change these permissions on the Express Server machine. Before you can use Administrator to configure databases, check with your system administrator to make sure you can access XPDDDATA.DB in read/write mode. The other database files can be changed to read-only.

Your system administrator must also specify the database file path in the ExpressServerDBPath configuration setting on the Express Server machine. Otherwise, neither Administrator nor the client applications will be able to connect to the Express Server machine.

If an Administrator or client application user connects to Express Server using security, Administrator or the client application will first look for the location of the application databases in the path setup for the account used. If an Administrator or client application user connects to Express Server *without* using security, the default user ID is used. If that user ID has a path setup, Administrator or the client application will use this as its “home” directory.

The next section describes how to set .INI file options. You will specify whether to connect to Express Server with or without security by setting the .INI file options for Administrator (in the XADMIN.INI file) or a client application (in the .INI file for that client application).

Setting .INI File Options

To provide connections to Express Server, you must edit the options in the [Express] section of the client application and the Administrator .INI files. Table 2.1 describes the options you can set. The subsections following the table provide additional information on and examples of the option settings.

Table 2.1 .INI Options for Express Server Connections

Option	Description	Settings
ServerDefault	Specifies the default connection of the Express Server connection you want to make. You will usually specify a setting for this option when the ServerPrompt option is set to 0.	An integer corresponding to the <i>n</i> in the ServerString <i>n</i> option. (Default is 0.)
ServerPrompt	Specifies whether to prompt, with the Connect dialog box, for which kind of Express to start.	0 = Do not prompt for which kind of Express to start. (Default.) -1 = Prompt for which kind of Express to start.
ServerType	Specifies whether to connect to a local or remote machine. You will usually specify a setting for this option when the ServerPrompt option is set to 0.	0 = Local (Default.) 1 = Remote
ServerDescription <i>n</i>	Specifies a description for an Express Server connection. The <i>n</i> is an integer you specify to identify different connections. Specify a description for each ServerString <i>n</i> specification.	A string describing the Express Server connection. If the string contains more than about 40 characters, the text is truncated when it is displayed. (No default.)
ServerString <i>n</i>	Specifies a string binding used to make an Express Server connection. The <i>n</i> is an integer you specify to identify different connections.	The string binding used to connect to Express Server. (No default.)

Table 2.1 .INI Options for Express Server Connections

Option	Description	Settings
ServerVersion n	Specifies which version of Express Server is running. The n is an integer you specify to identify different connections.	0 = Express Server 5.x (Default) 1 = Express Server 6.x
ServerLogin n	Specifies whether to prompt, in the Server Login dialog box, for which kind of authentication to provide. The values of -2, 1, 2, 3, 4, and 5 are only valid when the ServerVersion option is set to 1. If the value is -2 or from 1 to 5, you must also include the DomainName option in your .INI file for each ServerString. The n is an integer you specify to identify different connections.	Host authentication: -2 = For domain login -1 = For machine login 0 = No authentication (Default.) Domain authentication: 1 = At CONNECT level 2 = At CALL level 3 = At PACKET level 4 = At PACKET INTEGRITY level 5 = At PACKET PRIVACY level
DomainName n	If the ServerLogin value is -2, 1, 2, 3, 4, or 5, then you must include this option for each server string. Specifies the last domain name given by the user for this ServerString. The n is an integer you specify to identify different connections. It is not mandatory to specify a setting for this option. If you do not specify a setting, the login dialog will display an empty string.	A string describing the domain name. (No default.)
UserID n	The last user ID given by the user for this ServerString. The n is an integer you specify to identify different connections.	A string describing the user ID. (No default.)

The following sections describe:

- Which of the above options you *must* add to your .INI file
- Setting up your .INI file for one service versus multiple services of Express Server on the same machine

- Connecting to one remote Express Server machine all the time *or* choosing from two or more remote machines
- Taking advantage of the security options
- Running Administrator, a client application, or Personal Express™ on the same machine as Express Server

Specifying the Express Server Version

You must specify the version of Express Server to which you want to connect. To do so, add the `ServerVersion $n$` option to the [Express] section of the .INI file, as shown in the following example. Set this option to 1 to specify a connection to Express Server 6.0.

```
[Express]
.
.
.
ServerVersion1=1
```

Notice that you must include an integer at the end of the option name to identify the connection, even if there is only one possible connection.

Specifying the String for Connecting

The `ServerString $n$` option you use depends on whether you are running one service of Express Server or multiple services of Express Server on a single machine.

If you are running one service of Express Server on one machine, the `ServerString $n$` option takes the form *protocol:host*.

If you are running multiple services of Express Server on one machine, the `ServerString $n$` option takes the form *uuid@protocol:host*.

You might need to contact your system administrator for assistance in specifying the appropriate setting. The following list describes the parts of the setting.

- *uuid* is the universal unique identifier specified in the Configuration Manager on the Express Server machine by the system administrator.

The uuid is required only if you have more than one service of Express Server running on the same machine.

- *protocol* is the RPC transport protocol offered by Express. This may be `ncacn_ip_tcp`, `ncacn_ip_spx`, or another RPC-supported protocol, depending on the network being used.
- *host* is the IP address or domain name of the Express Server machine.

If the service to which you want to connect has the default uuid, you do not have to specify the uuid. Also, if you want to connect to the service on the same machine and the direct call SNAPI module is loaded, for that service you do not need to specify a value for the server string.

After specifying a string binding, specify a description for that string in the `ServerDescriptionn` option. This option should contain a text string describing the Express Server machine. The string is displayed in the Connect dialog box when referring to that connection.

Supporting Multiple Express Server Connections

If you have a single Express Server machine to which you always want a user to connect, set the options in the Express section of the client application and Administrator .INI files similarly to those shown in the following example. .

```
[Express]
ServerType=1
ServerDefault=1
ServerDescription1=Express Server 6.0 on Caesar
ServerString1=ncacn_ip_tcp:caesar
ServerVersion=1
```

If a user needs to access Express Server on different machines at different times, you might want to prompt the user when connecting to Express at startup and allow the user to choose from multiple Express Server connections. You do so by setting `ServerPrompt` to -1. You also must specify the `ServerStringn` and `ServerDescriptionn` options (and optionally a `ServerLoginn` option as described in the next section) for each configured Express Server connection. The following lines show an example of an [Express] section, set up for multiple connections.

```
[Express]
ServerPrompt=-1
ServerType=1
ServerDefault=1
ServerDescription1=Express Server 6.0 on Octavius
ServerString1=ncacn_ip_tcp:octavius
ServerVersion1=1
ServerLogin1=-1

ServerDescription2=Express Server 6.0 on Antonius
ServerString2=ncacn_ip_tcp:antonius
ServerVersion2=1
ServerLogin2=0
```

These settings enable a user to choose from one of two descriptions, which will be displayed in the Connect dialog box. The default description is the Express Server machine specified by `ServerString $n$` , where n is the value of `ServerDefault`. In the example, notice the setting of the `ServerLogin1` option. If a user selected **Express Server 6.0 on Octavius** in the Connect dialog box, the Login dialog box is displayed before the product attempts to connect to the Express Server machine. If the user selects **Express Server 6.0 on Antonius**, the Login dialog box is not displayed.

Each database on Express Server supports one read/write database attachment and multiple read-only database attachments. Administrator requires a read/write database attachment. Client applications will attach a database as read-only, unless you set that database's `ReadOnly` property to "No," which allows a client application to attach your database as read/write.

Taking Advantage of the Security Options

This section describes the type of security that is available and how you can take advantage of it.

There are three levels of authentication you can specify: domain, host, and none.

Domain authentication is the preferred way of authenticating clients for Windows NT Server, and also for Express Server 6.0. It is highly secure and provides different levels of authentication. Express Server can set a

minimum level of authentication for all its clients. If you want to use domain authentication, you must first meet the following requirements:

- The client and the server have to be on the Microsoft network
- The user has to have a domain account within the same domain that the server machine is on

Host authentication should be used when the client machine is not on the Microsoft network. Host authentication provides only one level of authentication. However, it can provide two different kinds of login:

- **Domain login** - Use when the server machine is within the domain and the user has a domain account.
- **Server machine login** - Use when the server machine is not in the domain (and/or if there is no domain account) and the user has a server machine account.

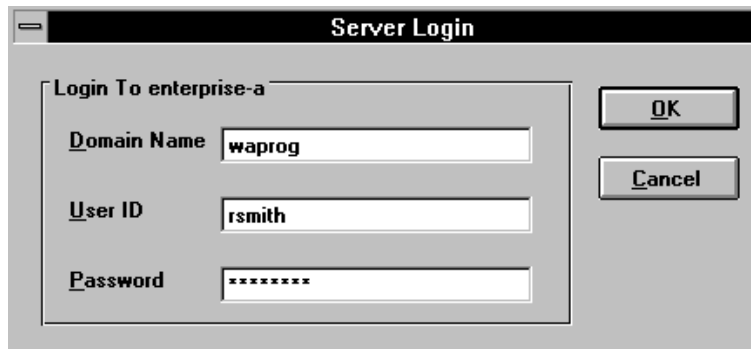
Alternatively, you can specify no authentication. In this case, unauthenticated clients will assume a default NT identity, which is read from the NT server registry.

To take advantage of the security options, include the `ServerLoginn`, `UserIDn`, and `DomainNamen` options in your client .INI file. The setting you specify for `ServerLoginn` determines:

- The kind of authentication (domain, host, or none)
- If you specify domain authentication, the level at which to prompt for authentication
- If you specify host authentication, whether to prompt for authentication at domain or machine login

Table 2.1 describes the settings for `ServerLoginn` in detail.

When you run Administrator and the client applications and attempt a connection to Express Server, you will be prompted for authentication, for example, as shown below by the Server Login dialog box for domain login.



In this case, the string “waprog” is stored in the .INI file as the setting for the DomainName option, and “rsmith” is stored as the setting for the UserID option.

The following example shows an [Express] section, set up for multiple connections.

```
[Express]
; general settings
ServerType=1
ServerPrompt=-1

; general settings for remote connection(ServerString) settings
ServerDefault=1

; First connection(ServerString) setting (Host Authentication)
ServerDescription1=Express Server 5.0 on olaphp1
ServerString1=b0973fa0-511a-11cf-8976-08000984b10d@ncacn_ip_tcp:olaphp1
ServerVersion1=0
ServerLogin1=-1
UserID1=rich

; Second connection(ServerString) setting (Domain Authentication)
ServerDescription2=Express Server 6.0 on enterprise-a
ServerString2=ncacn_ip_tcp:enterprise-a
ServerVersion2=1
ServerLogin2=1
DomainName2=waprog
UserID2=rsmith
```

These settings let a user choose from one of two descriptions, which will be displayed in the Connect dialog box. If the user chooses the first setting, the user is prompted to supply a user ID before being allowed to log onto the machine. If the user chooses the second setting, the user is prompted to supply both a user ID and a domain name before being allowed to connect.

Running Administrator and Express Server on the Same Machine

When Administrator or an Express Client product is installed on the same machine as Express Server, the way it connects to Express Server depends on whether you have set the `ServerStringn` setting in the Administrator or client application .INI file.

- If `ServerStringn` is unset, Administrator or the client application uses Local SNAPI to connect to Express Server.
- If `ServerStringn` is set to any value, Administrator or the client application uses Remote SNAPI to connect to Express Server.

As mentioned earlier in this chapter, you must install Remote SNAPI on the system that contains Administrator or a client application in order to connect to Express Server. But if you choose to assign a value to `ServerStringn`, and you run Administrator or a client application on the same machine as Express Server, you must install Local SNAPI instead.

In order to make a Local SNAPI connection, the server must have the Direct Call module installed. Refer to the *Oracle Express API Guide* for more information on Local SNAPI. The following INI file settings show how to connect to the Express Server using either Local SNAPI or Remote SNAPI.

```
ServerDescription1=Express Server 6.0 on Antonius (Local SNAPI)
ServerString1=
ServerVersion1=1
```

```
ServerDescription2=Express Server 6.0 on Antonius (Remote SNAPI)
ServerString2=ncacn_ip_tcp:antonius
ServerVersion2=1
```

If the server exposes several services, you can access only the default service using Local SNAPI.

You must install TCP/IP in order to make a Remote SNAPI connection.

Running Express Server and Personal Express on the Same Machine

This section describes how to run both Express Server 6.0 and Personal Express 5.0.1 on the same NT machine. Even though this configuration is not common, this section gives you the configuration information in the rare case you might need it.

As mentioned in other chapters in this manual, Personal Express 5.0.1 is available separately from Oracle.

If you need to run Personal Express on the same machine as Express Server, you must update the registry before you can switch from one version of Express to the other. SNAPI only keeps one registry setting for its local communication protocols (Direct for Express Server and DDE for Personal Express). As a result, the last-installed protocol DLLs will be the ones registered, and therefore used by Administrator, the Excel Add-In, or any other Express Client product. When switching from Express Server to Personal Express or vice versa, you must change the SNAPI DLL path in the machine's registry to point to the correct DLLs, as shown in the following example.

HKEY_CLASSES_ROOT\Oracle\OLAP\SNAPI\6.0.0\Local\DllPath

Note: If the client uses Remote SNAPI to connect to Express Server, you do not need to make any changes in the registry.

Chapter 3

Getting Started

Once Express Server is installed, you can install Administrator. You can then start Administrator and connect to Express Server. Once connected, you can create, configure, and maintain the databases that reside on the remote computer with Express Server.

This chapter explains how to get started with Administrator. It contains the following sections:

- What is Administrator?
- Starting and exiting Administrator
- Understanding the main window
- Summary

Note: This manual is intended to give you the basics in using Administrator, as well as more detailed information about Express database administration. For your convenience, Administrator provides you with an extensive Help system. As you work with Administrator, the Help system gives you a quick and easy way to find (and print) detailed information about Administrator features and step-by-step procedures. To help get you started using Administrator Help, this manual often suggests the Help topics you can refer to for more in-depth information.

What Is Administrator?

Administrator is a utility that provides a convenient interface for creating Express databases and configuring them for use with a client application.

This manual tells you how to use Administrator to perform database administration tasks. You will do most or all of your work using Administrator. In some cases, using the Express programming language gives you additional flexibility and power in performing certain tasks.

Using Administrator Versus Using Express

Express Server has a powerful programming language, hereinafter referred to as “Express.”

Most of the time, using Administrator is the easiest and fastest way to perform database administration tasks. You can also use Express commands to perform database tasks. You can write an Express program or enter Express commands in the Express Command window available in Administrator.

The advantage of using Administrator to perform database administration tasks is ease of use. The advantage of using Express commands is greater flexibility.

When writing and testing programs, you can perform simple debugging tasks in Administrator. For more complex debugging tasks such as stepping through code, you can use Personal Express 5.0, which is available from Oracle.

This manual focuses on using Administrator and mentions using Express to perform some tasks. For more information about using Express commands, see the *Express Language Reference*. For more information about writing Express programs, refer to the *Express Language Programming Guide*. The Administrator Help menu provides online Help for both Administrator and the Express Language.

Your Users' Client Application

Your users will be using one of the following types of client applications:

- A prebuilt Express client application available from Oracle
- A client application that you build yourself

As described in Chapter 2, Oracle client applications include the Excel Add-In and the Express Client products (Oracle Express Analyzer and Oracle Express Objects).

If your users use an Oracle client application with Express Server 6.0, the client application takes advantage of the data structures that are generated by Administrator. For example, when you configure your database, Administrator generates data structures that the Oracle client application uses to display short and long labels for the database objects you create.

The Excel Add-In and the other client applications do not require that you perform any of the configuration procedures described in this manual, but taking these steps makes the data in your databases easier to use and more attractive when displayed. Configuring your databases also lets your users take advantage of prebuilt client application features, such as drilling.

Alternatively, you can use SNAPI to write your own client application to use with Express Server.



See the *Oracle Express Server API Guide* for information about SNAPI and how to use it to develop client applications that interact with Express Server.

Like the prebuilt client applications, your client application can take advantage of the data structures generated by Administrator. Whether or not you want to use these data structures is up to you.

In examples, this manual generally refers to the Excel Add-In, because it is provided with Express Server 6.0. However, you can use other client applications in most cases, as well.

The Data Dictionary

The Data Dictionary is a collection of information about configured databases and their dimensions and measures. The Data Dictionary includes the data structures generated by Administrator, which the client application uses to organize data and to present it more clearly.

For example, in a configured database, the Data Dictionary contains the hierarchical information that allows client users to drill down from more aggregate dimension values (such as “Eastern Region”) to greater levels of detail (such as “Boston”).

With Administrator, you can create hierarchies and set properties for dimensions and measures. You can create long and short labels for dimension values. You can add language support. You can also set up an Express database that you can use for dynamic SQL reach-through. This offers the ability to obtain data directly from a relational database as users request the data.

Administrator makes it easy for you to create new Express databases. Administrator uses dialog boxes to help you create objects such as dimensions, variables, and formulas. Dialog boxes guide you through importing data from other files, such as ASCII or relational database files.

Advantages of a Configured Database

Oracle Express client applications can work with any Express database, because they are programmed to access dimensions, variables, relations, and formulas. However, you can customize databases to optimize the way your client application displays data.

If the database has not been configured with the procedures presented in this manual, the client application uses default values for the settings described here. The following are a few of the characteristics you can customize:

- The list of measures accessible to the client application user
- The ways the client application will handle labels and other display attributes of the data

- Saved selections of dimension values that provide customized subsets of the data to be displayed
- Hierarchy definitions for dimensions, which allow users to view, select, and drill data with reference to aggregate groupings

Configuring a database allows client application users to perform more sophisticated analyses on the data. In the Excel Add-In and other Express client applications, the advantages are most apparent in the Selector, where the user selects from lists of measures and dimension values. Using the configuration procedures, you can provide descriptive names for measures and dimensions, saved selections of dimension values, and dimension hierarchies. These names make it easier for users to analyze or work with the data.

In Express client applications *other than* the Excel Add-In, the advantages are also apparent in tables and graphs, where the user sees data values. You can specify the formatting characteristics for numeric data (e.g., the format for negative numbers and the currency symbol to use) and the way users drill to higher and lower levels of detail.

When you configure a database, information about that configuration is automatically stored in application database files that are provided with Express Server 6.0. You must make these database files available to each client application user who needs access to the user database you configure.

A database called XPDDDATA.DB is especially important for configured databases; it contains data structures that hold information specific to the databases you configure with Administrator, as described in the next section.

Databases You Should Know About

The Express Server 6.0 CD contains four *application database* files (XPDB.DB, XPDDCODE.DB, XPDDDATA.DB, and XPADMIN.DB), which are automatically installed on the same machine as Express Server when it is installed. These database files are necessary for connecting to Express Server and configuring user databases.

When you configure a database for a client application, Administrator records information about the database in the Data Dictionary. Specifically, Administrator updates data structures in XPDDDATA.DB. Thus, there is a special relationship between a configured database and the particular copy of XPDDDATA.DB that you used for the configuration. Users who access your configured database should be accessing the copy of XPDDDATA.DB associated with that database.

Two demonstration databases, DEMO.DB and XADEMO.DB, are also automatically installed on the same machine as Express Server.



For information about these database files, see Chapter 2. For information about ensuring that the client application can find these databases, see Chapter 5.

Starting and Exiting Administrator

Once you have installed Administrator, you can easily run it. Use the setup program to install Administrator. For more information, refer to Appendix A, “Installing Administrator.”



To start Administrator, double-click on the Administrator program icon.

Depending on how Administrator is set up, you might be prompted to select a connection. You must edit your XADMIN.INI file to allow a connection to Express Server 6.0; it will not work, otherwise. See Chapter 2 for information on how to connect to Express Server.

Once Administrator starts, you see the Main Window (Figure 3.1). It generally does not display database information until you create or attach a database.

Before starting Administrator, ensure that the file attributes of XPDDDATA.DB and your data database are read/write. (A data database is a database you create using Administrator.) Administrator will not start if XPDDDATA.DB is not read/write. After configuring a database and exiting Administrator, you can set XPDDDATA.DB to read-only.

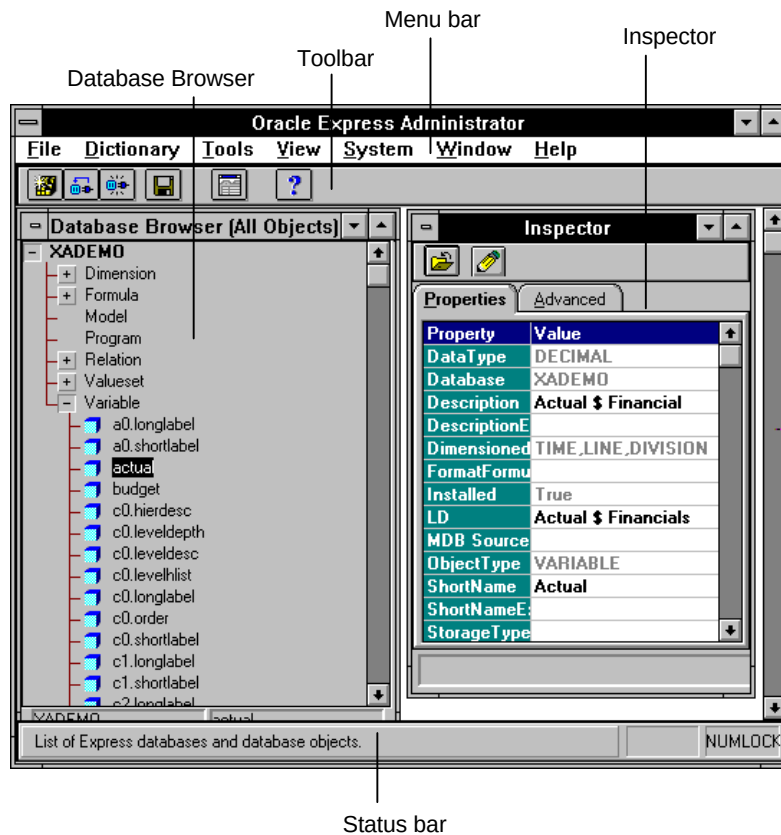


To exit Administrator, in the Main Window choose **Exit** from the File menu. When you exit, Administrator will terminate its connection to Express.

Understanding the Main Window

The Administrator Main Window provides access to the tools and information you use to create and configure databases. It contains a menu bar, a toolbar, the Database Browser, and a status bar. The Database Browser shows user-visible objects by default. You can also display the Inspector in the Main Window, and you can use the right mouse button to display popup menus.

Figure 3.1 shows the Administrator Main Window with the Express XADEMO database attached and the Database Browser and the Inspector displayed.

**Figure 3.1** Administrator Main Window

The next sections describe the following parts of the Main Window and how to get information while you work:

- Menu bar menus
- Toolbar
- Database Browser
- Inspector
- Popup menus

Menus

The menu bar on the Administrator Main Window gives you access to several menus. Table 3.1 provides an overview of these menus.

Table 3.1 Administrator Menus

Menu	Used to Accomplish These Tasks
File	Create, attach, and detach databases Install and uninstall databases Set properties for databases Set the language used in a database Convert application databases Connect to and disconnect from Express Exit Administrator
Dictionary	Define, edit, copy, rename, and delete Express objects Create saved selections Execute Express programs Set up default relations Specify search criteria
Tools	Import and export data Connect to and set up SQL data Write and execute Express commands
View	Specify whether Administrator displays the Database Browser showing all objects, user-visible objects, or search results
System	Specify a language for the system Specify a program to be executed at shutdown
Window	Arrange the layout of the Database Browser and the Inspector Arrange a minimized Database Browser and Inspector
Help	Access Administrator and Express Language Help systems



If you need more information about a menu as you work in Administrator, search for the following topics in the Administrator Help system:

- “Dictionary Menu”
- “File Menu”
- “Help Menu”
- “System Menu”
- “Tools Menu”
- “View Menu”
- “Window Menu”

Toolbar

The buttons on the Administrator toolbar provide shortcuts for frequently used actions. Table 3.2 shows each button on the toolbar and its action.

Table 3.2 Toolbar Buttons and Their Actions

Button	Purpose	Menu Equivalent
	Create a new database	Choose New Database from the File menu
	Attach an existing database	Choose Attach Database from the File menu
	Detach an existing database	Choose Detach Database from the File menu
	Display the Inspector	Choose Inspector from the Window menu
	Display the Help system contents screen	Choose Contents from the Help menu



For detailed information about these buttons, search for the following topics in the Administrator Help system:

- “Attaching Databases”
- “Creating New Express Databases”
- “Detaching Databases”
- “Inspector”
- “Main Toolbar”

Database Browser

You can display the Database Browser in one of three states. It can show all objects, only user-visible objects, or the results of a search operation. Use the View menu to specify which state is currently shown.

With any state, the Database Browser displays objects defined in all attached databases, arranged according to object type. Figure 3.2 shows the Database Browser displaying all objects in the Express XADEMO database.

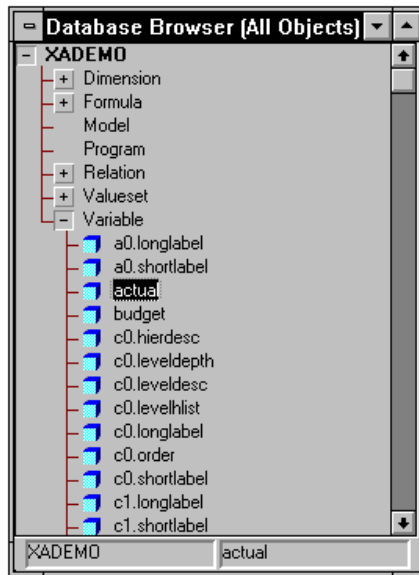


Figure 3.2 Database Browser, displaying all objects in the Express XADemo database.

The plus and minus signs next to the object types let you control how much detail you see in the Database Browser:

- Click on the plus sign next to an object type to display all database objects of that type.
- Click on the minus sign to display only the object type and not the individual objects.

After you apply search criteria in the Search for Database Objects dialog box (available from the Dictionary menu), the Database Browser displays only the objects that meet the search criteria.

Figure 3.3 shows the Database Browser displaying the objects that are dimensioned by Line in the Express XADemo database.

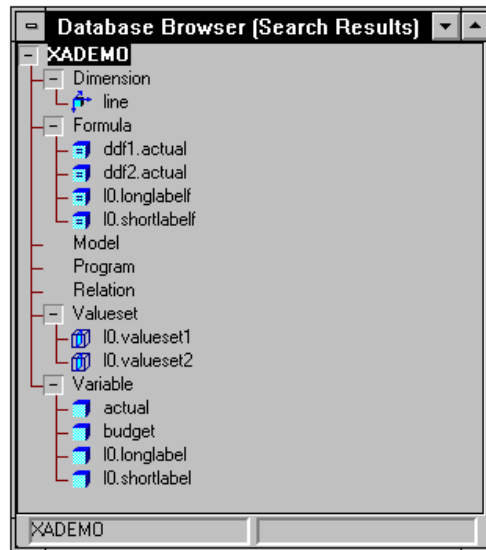


Figure 3.3 Database Browser, displaying the objects in the Express XADEMO database that are dimensioned by the Line dimension.

Note: The Database Browser showing search results is not available immediately after you attach or detach a database. To redisplay the Database Browser after you attach or detach a database, specify the search criteria again.



For detailed information about searching for objects, search for the following topics in the Administrator Help system:

- “Database Browser”
- “Search for Database Objects Dialog Box”
- “Searching for Database Objects”

Inspector

The Inspector displays the properties of a database object. A property specifies an attribute of an object.

You can use the Inspector to change the settings of many properties. You set some properties in dialog boxes when you create or edit objects. Some properties are read-only in the Inspector. You cannot set read-only properties; Administrator sets their values.

There are two tabs in the Inspector: the Properties tab and the Advanced tab. The Properties tab shows current settings of the object's properties, and the Advanced tab shows information about objects that are stored in the database. The Advanced tab shows statistics only for variables and dimensions. For other Express objects, the Advanced tab is dimmed.



To display the Inspector:

From the Window menu, choose **Inspector**.

or

Click on the Inspector button on the toolbar.

With the Inspector open, select an object in the Database Browser. The properties of the selected object appear in the Inspector.

Figure 3.4 shows the Inspector displaying some of the properties of the Geography dimension. The values of read-only properties are dimmed. The scroll bar along the right edge of the Inspector allows you to view properties that do not appear on the screen. You can also resize the Inspector.

The **Close** button closes the Inspector. The **Edit** button displays the Edit dialog box for the type of the currently selected object.

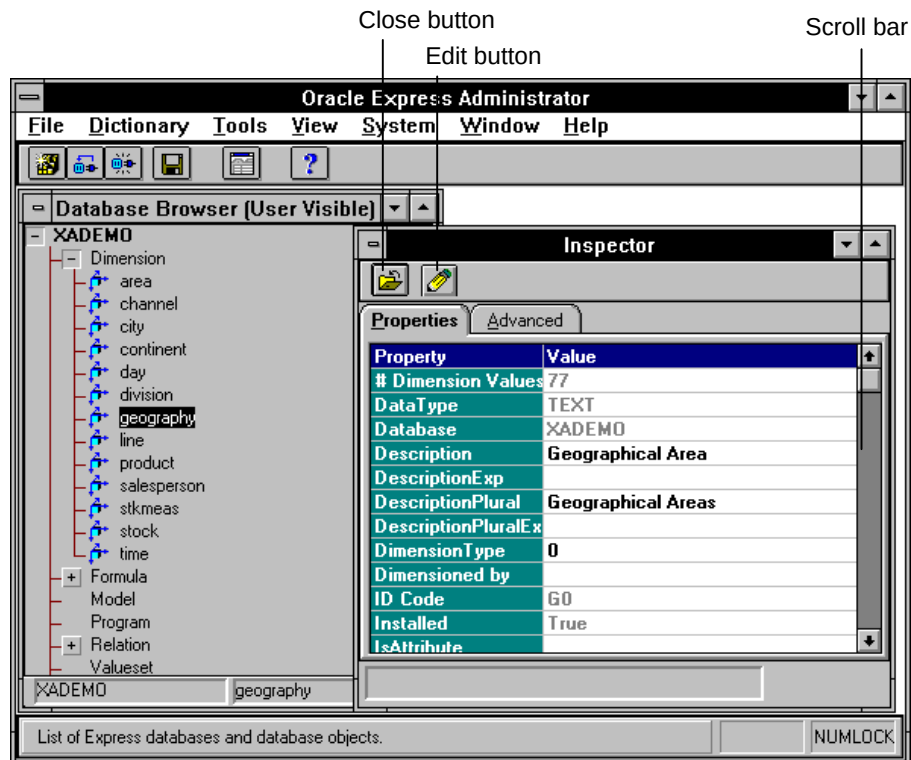


Figure 3.4 The Inspector, displaying properties of the Geography dimension.

When you inspect dimensions or variables, the Advanced tab shows read-only statistics about the selected object. When you click on the Advanced tab, Administrator calculates these statistics. On a large dimension or variable, this calculation may take some time. Figure 3.5 shows statistics for the Geography dimension.

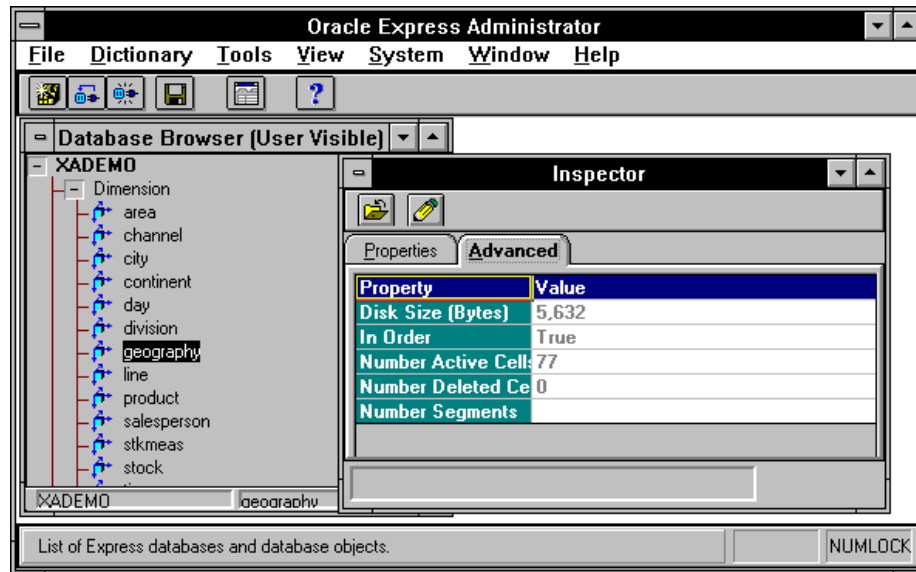


Figure 3.5 Statistics for the Geography dimension appear on the Advanced tab.



See Chapter 5 for information about setting properties.



For more information, search for the topic about the property you want to set or for the following topics in the Administrator Help system:

- “Editing Formulas”
- “Editing Programs and Models”
- “Editing Relations”
- “Editing Variables”
- “Inspector”
- “Installing Express Databases”
- “Setting Object Properties”

Using Popup Menus

In addition to using the menus in the Main Window menu bar, you can display the Dictionary menu by pressing the right mouse button.



To use the popup menu:

1. Move the mouse pointer to the Database Browser.
2. For executing, editing, copying, deleting, and renaming an object, select the object you want to act on by clicking on it.
3. Click the right mouse button to display the menu.
4. Click on the menu choice you want.



For more information about using the items on the Dictionary menu, search for the following topics in the Administrator Help system:

- “Copying Database Objects”
- “Defining Saved Selections”
- “Deleting Database Objects”
- “Dictionary Menu”
- “Executing Programs and Models”
- “Identifying Default Relations”
- “Renaming Database Objects”
- “Searching for Database Objects”

You can also search for topics on defining and editing database objects.

Getting Information While You Work

When you are working in Administrator, there are various ways to get assistance for tasks you want to perform:

- *The Help system* — You can access the Help system in several ways:
 - *Help menu* — From the Help menu on the main menu bar, choose **Contents, Search For Help On, Express Language Help 5.0**, or **Express Language Help 6.0**:

When you choose ..	You see ...
Contents	The main contents screen of the Administrator Help system
Search For Help On	The Search dialog box, from which you can search for topics on which you need help
Express Language 5.0 Help	The main contents screen of the Express Language (version 5.0) Help system
Express Language 6.0 Help	The main contents screen of the Express Language (version 6.0) Help system

- *F1 key* — View Help for the current item by pressing the F1 key. This is called *context-sensitive* help. For example, when you press F1 in a dialog box, you see the Help topic that describes that particular dialog box.

Note: The Help System also has a glossary that you can access from the *Glossary* button at the top of the Help window.

- *Status bar* - The status bar at the bottom of the Main Window displays useful information, which varies depending on your task.
- *Mouse pointer* - The appearance of the mouse pointer changes, depending on the task you are doing. Some mouse pointers are standard in Windows; for example, when you see the “wait” pointer, shaped like an hourglass, you know the system is busy and you must wait for it to finish before you can continue.

Chapter 4

Creating a Database

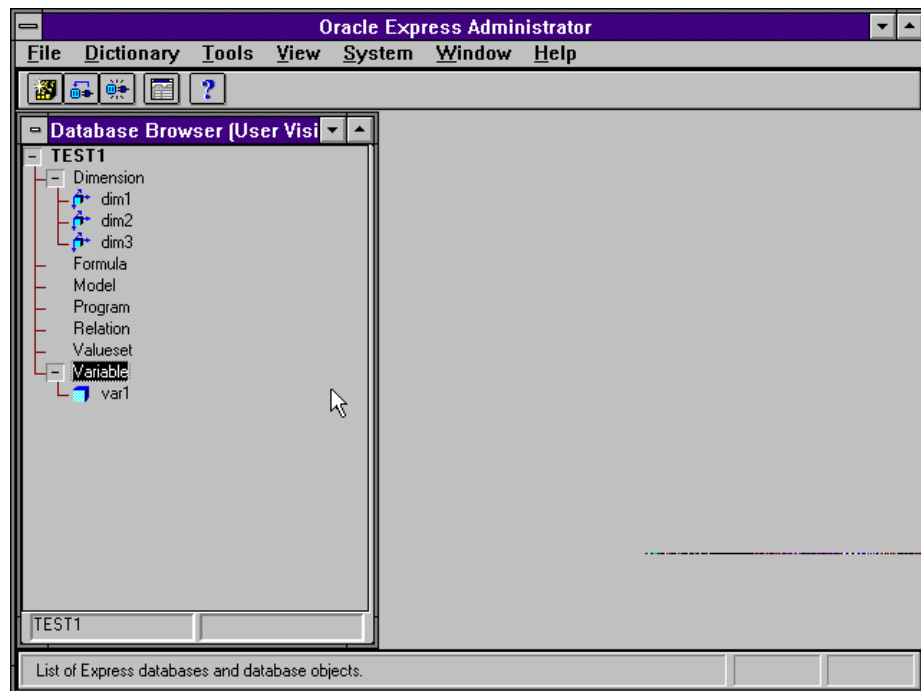
When you build a database from scratch, you need to create a new Express database. This chapter describes:

- How data is organized in the database dictionary
- How to create a new database
- Summary

The Database Dictionary

The *database dictionary* is Express Server's index to your database. The dictionary contains the names and descriptions of all your database objects. Express Server maintains it automatically when you add, delete, or rename objects.

The Database Browser displays the names of the objects in the database dictionary. The objects are grouped according to their type, as illustrated in the following figure.



By default, the Database Browser displays the objects that you specify as user-visible when (or after) you define them. To display both user-visible and non-user-visible objects, choose **All Objects** from the View menu.

Creating a New Database

When you create a new database, you will perform most or all of the following tasks:

- Create the database
- Define the database objects
- Add values to the dimensions you define
- Add data to the variables you define
- Add values to any other database objects you define

The following sections describe how to do each of these tasks.

Creating the Database

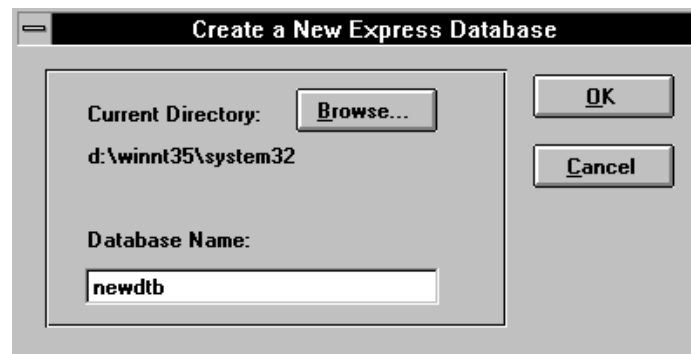
Your first step is to create a new Express database in which to store data.



To define a new database:

1. From the File menu, choose **New Database**. The Create a New Express Database dialog box appears.

In the Database Name box, type in a base file name of up to eight characters. Do not include a period (.) as one of the eight characters in the file name. Administrator adds the “.DB” extension to the file name.



The database will be created in the directory specified in the dialog box. Choose **Browse** to see the Browse dialog box if you want to select a different directory or a different drive or server.

2. Choose **OK**. The new database will be listed in the Database Browser.

Defining Database Objects

Once you have created a database and decided what objects you need to create, your next step is to define those objects.

First, you should define all of the dimensions you need.

Defining and Adding Values to Dimensions

You must define dimensions before you can define measures (variables, formulas, and relations). Remember that a dimension is a list of values that

provides an index to your data. Dimension values let you identify your data and select subsets of data with which to work. Some typical dimensions you might create are time, product, market, and region dimensions. For more information about dimensions, refer to Chapter 1.



To define a new dimension:

1. From the Dictionary menu, choose **Define**. From the popup menu, choose **Dimension** to display the Define a Dimension dialog box.
2. In the Name box, type a valid Express object name for the new dimension.

The screenshot shows the 'Define a Dimension' dialog box. At the top, the title bar reads 'Define a Dimension'. Below the title bar, there is a 'Name:' label followed by a text box containing 'geography'. To the right of the text box are two buttons: 'Close' and 'Define'. Below the 'Name' field, there are two tabs: 'General' (selected) and 'Labels'. The 'General' tab contains a 'Data Type:' label followed by a dropdown menu showing 'Text'. Below this, there is a section titled 'Dimensions' which contains two empty list boxes labeled 'Available:' and 'Selected:'. Between these two list boxes are two buttons: '>>' and '<<'. To the right of the 'Dimensions' section is an 'Options' section with three checked checkboxes: 'User Visible', 'Install', and 'Hierarchical'. Below the 'Options' section is a 'Storage' section with a 'Width:' label followed by a text box containing '12'. At the bottom of the dialog box, there is a 'Database:' label followed by a dropdown menu showing 'NEWDTB'.

The name must consist of 1 to 16 characters. For information about naming restrictions, see the entry for the DEFINE command in the Express Language Help.

3. In the Database box, select the database in which you want to define the new dimension.
4. In the General tab, select a data type in the Data Type box. You can select **Text**, **ID**, **Integer**, **Day**, **Week**, **Month**, **Quarter**, **Year**, or **Conjoint**.

- If you select **Conjoint**, select each base dimension from the Available list, beginning with the fastest-varying dimension. Then choose the right arrow button to add the dimension to the Selected list, or just double-click on the dimension. For more information about conjoint dimensions, see Appendix B.
 - If you select **Text** and all dimension values will have about the same number of characters, type the maximum number of characters for a dimension value in the Width box. Longer values will be truncated. Defining a fixed width where possible improves the efficiency of your database.
5. To list the dimension in the Database Browser and Selector in Express Client products, select the **User Visible** box. To create the supporting database objects for Express Client products, select the **Install** box.
 6. To define a hierarchy for the dimension, select the **Hierarchical** box.
 7. In the Labels tab, type a description for the dimension in the LD box. The description can include spaces and provides internal documentation for the dimension. In Express Client products, the LD appears in the Object Inspector.

The screenshot shows the 'Define a Dimension' dialog box with the 'Labels' tab selected. The 'Name' field contains 'geography'. The 'LD' field contains 'Geography Dimension Values'. The 'Short Name' field contains 'Geog. Area' and the 'Short Name (Plural)' field contains 'Geog. Areas'. The 'Description' field contains 'Geographical Area' and the 'Description (Plural)' field contains 'Geographical Areas'. The 'Database' dropdown at the bottom is set to 'NEWDTB'. Buttons for 'Close' and 'Define' are in the top right.

Define a Dimension	
Name:	geography
Close	
Define	
General Labels LD: Geography Dimension Values Short Name: Geog. Area Short Name (Plural): Geog. Areas Description: Geographical Area Description (Plural): Geographical Areas	

8. Type a short name for the dimension in the Short Name box, using any combination of upper- and lowercase letters. For example, the short

name for the TIME dimension might be “Time.” In Express Client products, the short name appears in tables, graphs, and the Dimension Bar.

9. Type a more descriptive name in the Description box. For example, the description for the TIME dimension might be “Time Period.” In Express Client products, the description appears in the Database Browser and in the Selector.
10. Type the plural forms of the names in the Short Name (Plural) and Description (Plural) boxes. For example, if the description is “Time Period,” the plural name is “Time Periods.” Express Client products use these names when displaying scripts in the Selector.
11. Choose **Define** to define the dimension and, for installed and hierarchical dimensions, to create the supporting database objects.
12. Choose **Close** when you have finished defining dimensions.

Additional database objects are automatically created to support installed and hierarchical dimensions. If you use a client application provided by Express, that client application will use these additional database objects.

After defining a dimension, you must provide values for it. You can do this by importing data from an external source or by typing in the values manually. The next section describes how to enter the values manually. Chapter 6 describes how to import data.



To add values to a dimension:

1. Double-click on a dimension in the Database Browser to display the Edit *Dimension* dialog box.
2. To add values to the dimension, select the Values tab.

Edit PRODUCT

Name: Close

General **Labels** **Values**

Child	Parent	Or
TOTALPROD		
AUDIODIV	TOTALPROD	
PORTAUDIO	AUDIODIV	
PORTCD	PORTAUDIO	
PORTST	PORTAUDIO	
PORTCAS	PORTAUDIO	
AUDIOCOMP	AUDIODIV	
TUNER	AUDIOCOMP	

Dimension Values

Add...
Delete...
Sort...

Hierarchy Info

Hierarchy: Add...
Description: Delete...
☒ Default Levels... Calculate...

Drill Direction:

Database:

3. Choose **Add** in the Dimension Values box. Depending on the data type of the dimension, you will see one of the following dialog boxes:
 - **Add Dimension Values dialog box** — Adds text dimension values. Select **Uppercase dimension values when adding** to convert dimension values to uppercase; otherwise, the dimension values are stored exactly the way you type them in. You do not need to use quotes around mixed-case values, as you would in Express. If you specified a maximum width when you defined the dimension, do not exceed that width when entering values. Type values in the appropriate boxes, then choose **Add**. When you are done adding values, choose **Close** to return to the Values tab of the Edit *Dimension* dialog box.
 - **Add Integer Dimension Values dialog** — Adds integer values. Type in the number of values you wish to add, and choose **OK**.
 - **Add Time Dimension Values dialog box** — Adds time periods. In the First Time Period box, type in the first value, then choose **Add**. To add more values, select **Add values before first time period** or

Add values after first time period. Then type in the number of periods you wish to add. Choose **Add**, then **Close**.

The new dimension values will appear in the Dimension Value or child column. For an installed dimension, the short names will appear in the Short Name column and the long names will appear in the Long Name column. You might need to use the horizontal scrollbars to view these columns.

4. To delete a dimension value, select the value in the table, then choose **Delete**. You will be asked to confirm that you want to delete the dimension value.
5. To edit an existing value, double-click on it. Then you can add and delete individual characters.
6. When you are finished making changes to the dimension, choose **Close**.



Refer to the DEFINE DIMENSION command in the Express Language Help system for more information about data types and defining dimensions.

Defining and Using Hierarchies

A dimension can have one or more hierarchies, which identify the values that contribute to subtotals and totals in the data. When Express Client product users look at a table, they can expand the hierarchy (“drill down”) to see the values that contributed to a total, or collapse the hierarchy (“drill up”) to see just the aggregate values. Whenever you add values to a hierarchical dimension, you must also maintain the hierarchy.

For example, suppose your product is sold worldwide. You can define a GEOGRAPHY dimension with four levels: cities, countries/regions, continents, and world. An example of a city would be TORONTO. It would belong to the CANADA country/region level, which would belong to the AMERICAS continent level. All continents would belong to the WORLD level. A SALES variable dimensioned by GEOGRAPHY might have data only at the cities level.

Typically, you will add data to a variable (that is dimensioned by a hierarchical dimension) for the lowest level of the hierarchy. For example,

in a variable dimensioned by GEOGRAPHY, you will add data for just the cities level. You can then *roll up* the data, which means that Express will calculate the sum for every level of the hierarchy. In other words, if you add sales figures for *cities only* to a variable, you can then roll up data in order to get sales figures for all other levels (the countries/regions level, the continents level, and the world level).

This section provides an overview of the following steps you take to define and use hierarchical dimensions.

1. Define a hierarchical dimension.
2. After adding values to the dimension, establish the parent-child relationship for the dimension values.
3. Calculate the hierarchy.
4. Define a variable that is dimensioned by the hierarchical dimension.
5. Add data to the lowest hierarchical levels in the variable.
6. Roll up the data for all levels of the hierarchy.

Step 1: Define a Hierarchical Dimension



While you are defining a dimension, you can specify that dimension as hierarchical. To do so, select the Install and Hierarchical boxes in the General tab of the Define a Dimension dialog box.

After you define the dimension, Express creates several objects to support the dimension hierarchy. Each object name's unique prefix consists of the first letter of the dimension and a digit, for example, "G1." One of these objects is a relation with the naming convention of *Xn.PARENT*. You will need this object later to roll up your data. Step 6 describes how to identify this object and use it.

You can also make an existing dimension hierarchical. To do so, double click on the dimension name in the Database Browser to open the Edit a Dimension dialog box, and select the Install and Hierarchical boxes in the General tab.

For more information about defining dimensions, see the previous section in this chapter entitled "Defining and Adding Values to Dimensions."

Step 2: Establish the Parent-Child Relationship

After defining the hierarchical dimension, you will add values to it, as described earlier in this chapter. Notice that on the Values tab in the Edit a Dimension dialog box, there is a child and a parent cell for each value you have added to the dimension. Every value is listed in the child column. The parent column is empty.



To assign a parent to a child:

1. Display the Values tab of the Edit a Dimension dialog box.
2. Click on the down arrow on the right of a parent cell. This displays a drop-down menu that lists every dimension value.
3. Click on the value in the drop-down menu that you want to assign as the parent of the child shown to the left of the parent cell. That value is displayed in the parent cell.

The following figure shows every parent that has been assigned to a child in the GEOGRAPHY dimension. Notice that the child at the highest level of the hierarchy, WORLD, has no parent.

Edit GEOGRAPHY

Name: Close

General **Labels** **Values**

Child	Parent	Or
WORLD		
AMERICAS	WORLD	
CANADA	AMERICAS	
TORONTO	CANADA	
MONTREAL	CANADA	
OTTAWA	CANADA	
VANCOUVER	CANADA	
EDMONTON	CANADA	

Dimension Values

Add...
Delete...
Sort...

Hierarchy Info

Hierarchy: Add...
 Description: Delete...
☒ Default Levels... Calculate...

Drill Direction:

Database:



For more information about assigning parent values to child values, search for the following topic in the Administrator Help system:

“Defining Hierarchies”

Step 3: Calculate the Hierarchy

When you finish assigning parent values to child values, you must calculate the hierarchy.



To calculate the hierarchy:

1. Choose **Calculate**. A dialog box prompts you to decide whether to calculate just the current hierarchy or all hierarchies for the current dimension. (You can create more than one hierarchy for a dimension.)
2. Choose **OK**.

This calculates the number of levels in the hierarchy, the order of child values, and other supporting structures.

3. Choose **Close**.



For more information about calculating hierarchies, search for the following topic in the Administrator Help system:

“Defining Hierarchies”

Step 4: Define a Variable Dimensioned by the Hierarchical Dimension

You will learn how to define a variable later in this chapter. When you define a variable, you specify the names of its dimensions.



To define a variable with a hierarchical dimension, select that dimension name when you define the variable.

For details about defining variables, see the section “Defining and Adding Values to Variables” later in this chapter.

Step 5: Add Data to the Lowest Hierarchical Levels in the Variable

This manual describes several ways to add data to your variable. What is important to understand is that, in most cases, you will probably add data to your variable for the lowest level of your hierarchy. In order to total that data for all other levels of the hierarchy, you must roll up that data, as described in Step 6.

For more information about adding data to your variables, see the section “Defining and Adding Values to Variables” later in this chapter. See Chapter 6 in this manual for details about importing data into a variable. See Chapter 9 for information about using the = and FILEREAD commands to assign data to a variable.



Also, search for the following topics in the Administrator Help system:

“Importing ASCII Data”
“Import ASCII Data Dialog Box”

Step 6: Roll Up the Data for All Levels of the Hierarchy

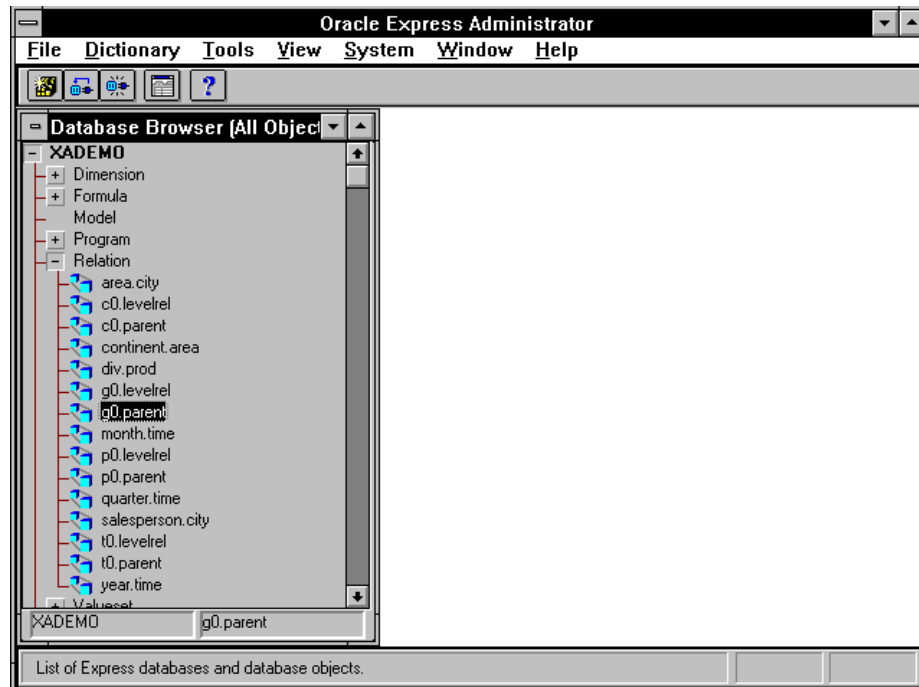
To roll up the data, you execute a ROLLUP command using the name of your variable, the name of the hierarchical dimension, and the name of a parent relation for that dimension that is automatically created by Administrator.

The following procedure describes how to get the name of the relation for your hierarchical dimension and how to use the ROLLUP command.

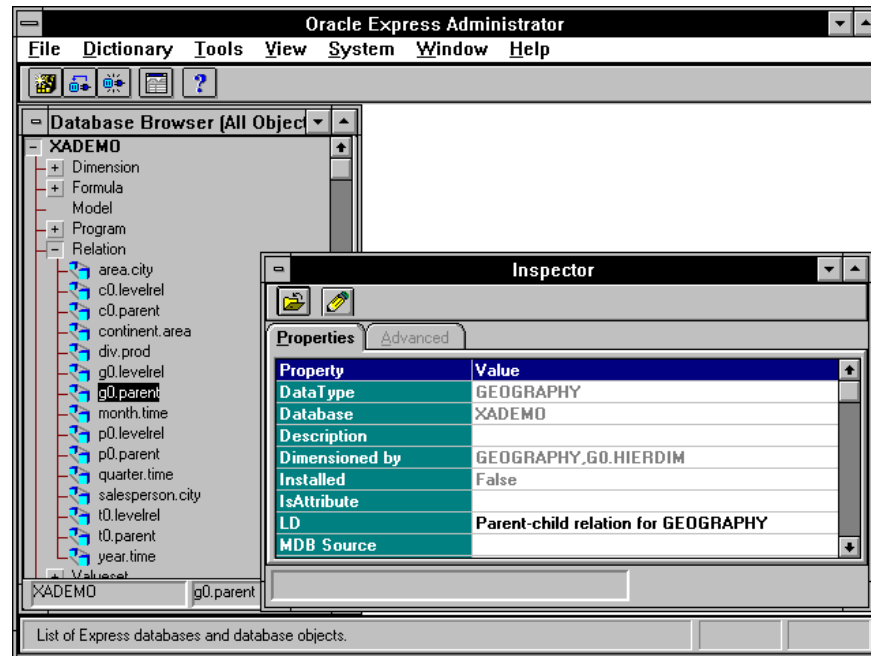


To roll up the data in your variable:

1. From the View menu, choose **All Objects**. This makes the name of every object in your database visible, including the supporting objects that are automatically created by Administrator.
2. In the Database Browser, click on **Relation** to display the names of all the relations in your database.
3. Select the $Xn.PARENT$ name for your dimension. Remember that the X will be the first letter of your dimension name, and the n will be a number. For example, in the following figure, GEOGRAPHY's $Xn.PARENT$ name is “G0.PARENT.”



4. From the Window menu, choose **Inspector**, if the Inspector is not already displayed.
5. In the Inspector, verify that the relation you have selected is the relation you need. Look for the LD property in the Inspector window. The property holds a text description of the relation. (You may need to resize the Inspector to read the entire description.) Its value will be "Parent-child relation for *dimension-name*." For example, the LD property for G0.PARENT would be "Parent-child relation for GEOGRAPHY." (Note that this property is read-write. The value described here is its default value, which may be changed.)



6. From the Tools menu, choose **Express Command** to display the Express Command window.

An alternative way to get the name of the parent relation is to use the OBJ function. In the upper window of the Express Command window, type the SHOW command with the OBJ function using the following syntax.

```
show obj(property 'PARENTREL' 'dimension-name')
```

For example, to get the parent relation name of the GEOGRAPHY dimension, you would type the following command, then choose **Execute** to execute the command.

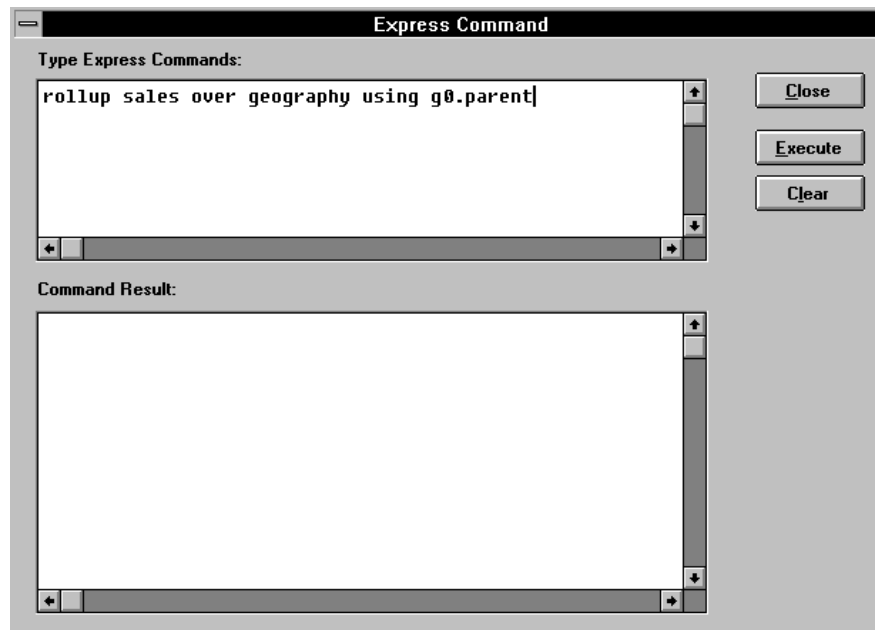
```
show obj(property 'PARENTREL' 'GEOGRAPHY')
```

7. In the upper window, type the ROLLUP command using the following syntax.

```
rollup variable-name over dimension-name using  
relation-name
```

So, for example, to roll up the SALES variable, you would type the following command.

```
rollup sales over geography using g0.parent
```



8. Select **Execute** to execute the command.

You can then close the Express Command window. If you open the Edit a Variable dialog box for the variable and look at the Data tab, you will see that data has been rolled up for every parent dimension value.

Whenever you add values to an existing hierarchical dimension, or you change or add data to your variable, you must roll up that data. It is a good idea to become familiar with both the ROLLUP and LIMIT commands. It may not always be necessary to roll up data for every dimension value. In some cases, it may be necessary to roll up data only for a certain level or branch of the hierarchy.



For more information, see the ROLLUP and LIMIT commands in the Express Language Help system.

See Chapter 8 for more detailed information about hierarchical dimensions, especially in regard to configuring your database.

Defining and Adding Values to Variables

Once you have defined and added values to dimensions, you can then create your variables. You can define variables for storing data that you import from an external source or enter manually.



To define a new variable:

1. Define all dimensions required by the variable.
2. From the Dictionary menu, choose **Define**. Then choose **Variable** to display the Define a Variable dialog box.
3. In the Name box, type a valid Express object name for the new variable.
4. In the Database box, select the database in which you want to define the new variable.
5. In the General tab, select a data type in the Data Type box. You can select **Boolean**, **Date**, **Decimal**, **ID**, **Integer**, **Shortdecimal**, **Shortinteger**, or **Text**.
6. In the Available box, select the fastest-varying dimension. Then choose the right arrow button to add that dimension to the Selected box. Repeat this step for each of the variable's dimensions, ending with the slowest-varying dimension.

Define a Variable

Name:

General **Labels** **Format**

Data Type:

Dimensions

Available: channel, city, continent, day, geography, product, salesperson, stkmeas, stock

Selected: time, line, division

Options

☒ User Visible
☒ Install

Storage

☐ Temporary
☒ Permanent
☐ In-Place

Width:

Sparse Dimensions

None

Database:

7. In the Options box, to list the variable in the Selector and Database Browser in Express Client products, select **User Visible**. To create the supporting database objects for Express Client products, select **Install**.
8. In the Storage box, select **Temporary** if you want the data to be discarded at the end of every session, or **Permanent** if you want to store the data from one session to the next.

Select **In-Place** only if you want to conserve disk space during data updates by writing new data to the same physical location as the old data. In-Place improves performance during data loads, but you lose the ability to discard a failed update. This option is available only for numeric data types. You can change the In-Place option at any time.

Type a value in the Width box if you want to restrict the number of bytes of storage for each value of the variable. A longer value will be truncated to this width. TEXT variables can have a width of 1 to 256 bytes. INTEGER and BOOLEAN variables can have a width of 1 byte.

Be sure that all integer values are between -128 and 127 before restricting the width to 1 byte.

9. In the Labels tab, type a description for the variable in the LD box. The description can include spaces and provides internal documentation. In Express Client products, it appears in the Object Inspector. You can change all of the names at any time by editing the variable.
10. Type a short name for the variable in the Short Name box, using any combination of upper- and lowercase letters. For example, the name for the V_SALES variable might be "Sales." Express Client products display the short name in tables, graphs, and the Dimension Bar.
11. Type a description in the Description box. If the short name is "Sales," the description might be "Sales Dollars." In Express Client products, the description appears in the Database Browser and the Selector.
12. For an installed variable with numeric data, you can specify its formatting.
13. If you want to create an unnamed composite, you can set sparse dimensions. An unnamed composite is similar to a conjoint in that it is intended for the efficient management of sparse data. You can do this only if Administrator is currently connected to Express Server 6.0 or later. See Chapter 2 for more information.
14. Choose **Define**.
15. When you are through defining variables, choose **Close**.

Once you have created a variable, you can add data to that variable. You can either enter data values manually or import data. This section describes how to enter data manually. See Chapter 6 to learn how to import data into a variable.



To enter data in a new variable:

1. Double-click on the variable in the Database Browser to display the Edit *Variable* dialog box.
2. To edit the data, select the Data tab, click on the field you want to edit, and enter a new value.

Edit ACTUAL

Name: Close

General **Labels** **Format** **Data** **Data Driven Format**

DIVISION ↓

	JAN95	FEB95	MAR95
LINE	ACTUAL	ACTUAL	ACTUAL
REVENUE	533,362.88	572,796.23	707,197.75
COGS	360,810.66	400,901.65	478,981.61
GROSS MARGIN	172,552.22	171,894.58	228,216.13
MARKETING	37,369.50	38,866.53	51,223.68
SELLING	89,007.38	86,458.00	102,232.29
R.D	24,307.50	23,399.72	39,942.01
OPR. INCOME	21,867.84	23,170.33	34,818.15
TAXES	15,970.39	16,319.38	23,029.30
NET INCOME	5,897.45	6,850.95	11,788.86

Database: ↓

If the variable has more than six dimensions, a message box appears saying you cannot view the data in Administrator, because the variable has more than six dimensions. (Use Oracle Express Objects to view and edit the data in a variable that has more than six dimensions. If you just want to view a small portion of the variable's data, see Chapter 9 for information about using the LIMIT and REPORT commands.)

Otherwise, you can use the scrollbars to see the full two-dimensional slice of data. To see additional pages of data, select different values for the other dimensions in the *Dimension* list boxes. You can enter new values or edit a value by double-clicking on it.

You can also rotate the data display by switching the position of the dimensions. For example, if the display shows time periods down the first column and line items on separate pages, you can switch them so that line items appear down the first column and time periods appear on separate pages. Just click on the dimension name and drag it onto

another dimension name. You can also drag the dimension values in the table. The dimensions will switch positions.

To change the width of the columns, click the right mouse button. Then choose **Column Width** to display the Column Width dialog box. Type in new values for the width of the columns.

3. For an installed variable with numeric data, you can specify its formatting.
4. For an installed variable, you can define its data-driven formatting formulas.
5. Choose **Close** to return to the Main Window.



Refer to the DEFINE VARIABLE command in the Express Language Help system for more information about defining variables.

Using Other Database Objects

You can also create formulas, models, programs, relations, and valuesets.

You should consider using formulas whenever possible. Formulas calculate temporary data that you do not need to store permanently in your database. For example, if sales are always units sold multiplied by the month's price, you can define a formula to calculate sales and avoid storing redundant data in the database. See Chapter 9 for more information about using formulas.

You will probably also want to use hierarchies. A hierarchy is *not* an object. Instead, a hierarchy is a relation that you create between dimensions. Refer to “Creating and Using Hierarchies” earlier in this chapter for more information about hierarchies.



For information about creating these Express database objects, search for the following topics in the Administrator Help system:

“Defining Formulas”
“Defining Hierarchies”
“Defining Models”
“Defining Programs”
“Defining Relations”
“Defining Valuesets”

Alternatives for Creating Time Dimensions

If you are creating time dimensions, you can use Express commands instead of the Administrator interface to take advantage of some special features Express offers. Keep in mind that the time dimensions you create with Administrator are the dimensions expected by the Excel Add-In and other Express Client products. For example, if you create time dimensions using Express commands, your users may not be able to take advantages of such features as drilling down in their client applications. However, if you find you need to create time dimensions other than the ones available in Administrator, refer to the DEFINE DIMENSION command in the Express Language Help for more information.

Upgrading Express 5.x Databases to Express 6.0

If you are a previous user of Express, you can upgrade your existing Express 5.x (or earlier) databases to Express 6.0.

This section describes upgrading with the following configurations:

- An Express Client product connected directly to Express Server 5.x using Remote SNAPI.
- An Express Client product connected to Personal Express only, either on a local PC or a local area network (LAN).

- An Express Client product connected to Personal Express communicating with Express Server 5.x on a remote machine via Express Communication Architecture (XCA).

How to Upgrade a Database to Express 6.0

The following procedure describes how to use Administrator to upgrade an Express 5.x (or earlier) database to Express Server 6.0.



To upgrade an Express 5.x database to Express 6.0:

1. Start Express 5.x (or earlier).
2. Attach the database you want to upgrade. See the DATABASE command in the Express Language Help system for more information.
3. At the Express command prompt, use Express commands to export your Express database to an EIF file. See the EXPORT command in the Express Language Help system for more information.
4. Move or copy the EIF file of your Express database to the server machine on which Express 6.0 is installed; for example, by using a file transfer program such as FTP.
5. Connect Administrator to Express 6.0.
6. Create a new database.
7. Import the EIF file into the new database. See Chapter 6 for information about importing an EIF file.

Tip: It is a good idea to limit the NAME dimension to “all,” as well as executing an ALLSTAT command before you export.

If you want to upgrade an Express database version 4.8 or later, you can use Administrator to export your database. In other words, start Express, connect the appropriate version of Administrator to Express, use Administrator to export the database to an EIF file, move the EIF file to the server on which Express 6.0 resides, connect Administrator to the server, create a new database, and import the EIF file. The following table describes which version of Administrator you can use with each version of Express.

Use ...	To Connect to ...
Administrator 1.0	Express 4.8
Administrator 1.1	Express 4.8.1
Administrator 2.0	Express 5.0
Administrator 2.0.1	Express 5.0.1

After you have imported your database into Express Server 6.0, you must make sure that Administrator or any other client application can find any data or metadata associated with that database. To do so, read whichever of the following sections is appropriate to your configuration.

Personal Express Only or a Remote SNAPI Connection to Express Server 5x

This section assumes you or your users have been using Personal Express (without making a connection to Express Server 5.x) *or* your client application previously connected directly to Express Server 5.x using Remote SNAPI.

In either case, you must use the previous upgrade procedure to upgrade the following databases to Express Server 6.0:

- XPDDDATA.DB
- Any databases you created that contain data
- Any databases you created that contain programs

Personal Express 5x XCA Connection to Express Server 5x

If you used a Personal Express 5.x XCA connection to Express Server 5.x, most, if not all, of the data was located on the Express Server 5.x machine. You may have a mirror image of the Express Server 5.x data in one or more Personal Express databases.

If you want to maintain an XCA connection between Personal Express 5.x and Express Server 6.0, you must simply upgrade your Express Server 5.x databases, using the previous procedure. The other pieces of your configuration can remain the same.

What to Do After the Upgrade

When you move databases (such as XPDDDATA.DB) to the machine on which Express Server 6.0 is installed, you can put them in the directory of your choice. However, you should then ask your system administrator to add that directory to the ExpressServerDBPath configuration setting on the Express Server machine. See Chapter 2 for information about the steps you take to ensure that Administrator and any client applications can connect to Express Server 6.0 and locate any databases on the server machine.

When you “upgrade” an existing Express database to Express 6.0, you are merely moving it to a new server. You can then take additional steps to take full advantage of the new features that Express 6.0 provides.

- You may want to convert your conjoint dimensions to composites so that Express automatically maintains them. Chapter 9 describes composites in detail. Appendix B describes the procedure for converting conjoint dimensions to composites and vice versa.
- Many of the commands in Express 5.x are now obsolete. If your Express 5.x database has any programs or models, be sure to check the list of commands that are not supported in Express 6.0. It is a good idea to compile every program and model after you import your database into Express 6.0; if your program or model contains obsolete commands, check the error messages that are displayed when you try to compile to find out which commands are no longer supported. See the appendix in the *Express Language Programming Guide* for the list of these commands.
- You may also want to take advantage of the easier way to create and manage hierarchical dimensions that Administrator provides, as described earlier in this chapter.

Chapter 5

Structuring Data

This chapter explains how to use Administrator to work with databases and to structure your data. It contains the following sections:

- Working with databases
- Working with dimensions
- Working with measures
- Executing Express commands
- Summary

Note: This chapter describes how to work with databases, dimensions, and measures (variables, relations, and formulas). For more information on these Express objects, see the Express Language Help system. For information on copying, renaming, and deleting Express objects, see Chapter 7.

Working with Databases

Administrator provides a simple way to maintain databases. You can create new databases in Administrator, or you can use Administrator to modify databases that were created in Express. You can also use Administrator to install (i.e., configure) databases for use with the client application.

The following sections describe how to:

- Attach an existing database
- Install a database for use with the client application
- Set properties for a database
- Convert an application database for use with the client application
- Work with languages
- Detach a database
- Ensure that the client application can find your databases

Attaching a Database

Once you or someone else has created an Express database that you want to work with (or if you want to try one of the demo databases), you can attach (or open) it in Administrator. Administrator does not allow you to attach a read-only database. You must have read/write access to the database file at the operating system level to work with the database in Administrator.

Once you attach a read/write database, its contents appear in the Database Browser, and you can edit the database objects. You can have a maximum of four databases attached simultaneously.

To attach an Express database:

1. From the File menu, choose **Attach Database** to display the Attach Database dialog box.
2. If the database is located on a different drive and directory from the current one, select the correct ones in the appropriate boxes.

If the database is located on a different server from the one to which you are currently connected, you must first disconnect from the current server, then connect to the server on which the database is located. (From the File menu, choose **Disconnect from Express**, then choose **Connect to Express** to use the Connect dialog box.) After you have connected to a different server, you can then display the Attach Database dialog box and select the correct drive and directory in the appropriate boxes. The Attach Dialog box only allows you to switch drives and directories for the current server.

3. Select the database from the list in the File Name box, or type in its name. If the database does not have a .DB file name extension, you can either type in the extension, such as *.EXP, or select **All Files** in the List Files of Type box. Choose **OK**.
4. If the database is protected by a password, you are prompted to enter the password.

Installing a Database

Client application users can attach any Express database, regardless of whether it has been set up using the installation procedures described in this chapter. However, you can customize the way client application briefings, projects, or queries handle databases by performing optional configuration tasks. When you perform these tasks, you make client application briefings, projects, or queries more powerful. See Chapter 1 for information on the advantages of configured databases.

The optional configuration tasks include the following:

- Customize the way the client application handles the dimensions and measures in a given database.
- Specify certain databases to be attached and programs to be run when the client application starts.

When you install a database, you do not automatically install any of its dimensions or measures. However, you do place the database's name in the Data Dictionary.



For information on installing databases, search for the following topic in the Administrator Help system:

“Installing Express Databases”

Setting Properties for a Database

You can specify the way the client application will handle a given database by specifying its *properties* in Administrator. The properties of a database are a group of its characteristics such as the path name, the access mode (read-only or read/write), and whether the database should be hidden from the client application user.



To view and set the properties of an installed database, from the File menu, choose **Setup**. Then choose **Properties** in the resulting dialog box.

Table 5.1 describes the properties of databases. All database properties are read/write. Search the Administrator Help system for more information on each property.

Table 5.1 Database Properties

Property	Description
Attach Order	The database's place in the attach order (relevant only if the database will be attached automatically)
AutoAttach	Whether the database should be attached automatically when the client application starts
Description	A phrase describing the database
Language Dimension	The dimension that holds the names of all the languages that the client application can use with the database
PathName	The full path name of the directory in which the database file resides, including the file name
Post-Attach Program	The name of an Express program that the client application runs after attaching the database
Post-Detach Program	The name of an Express program that the client application runs after detaching the database
Pre-Attach Program	The name of an Express program that the client application runs before attaching the database
Pre-Detach Program	The name of an Express program that the client application runs before detaching the database
ReadOnly	Whether a user can attach the database as read-only or read/write
User Data	Whether the database and its contents are visible to the user in the client application



For information on setting database properties, search for the following topic in the Administrator Help system:

“Installing Express Databases”

Working with Languages

To work with multiple languages in Administrator, you must understand how the client application handles languages. The client application itself can have one or more system languages, with one language being the current language. Each database also has a current language. You can use Administrator to specify languages.

System Languages

The default system language for the client application is ENU, for U.S. English. If you have another language that you want to use, you can add it to the system. When you no longer need the language, you can delete it. You add and delete system languages and specify the current language using the System Language Setup dialog box, which is shown in the following figure.



To display the System Language Setup dialog box, choose **Language** from the System menu.



Figure 5.1 System Language Setup dialog box in Administrator.

Database Languages

When you install a database, Administrator creates a language dimension for that database. The default name of the language dimension is `XXX.LANGDIM`, where `XXX` stands for the database name. By default, the dimension has a value, `ENU`, for U.S. English. You can add additional values for any other languages in the client application. As you configure dimensions and measures, you will find that several of their supporting data structures are dimensioned by the language dimension, so you can specify different values for different languages.

▶▶ To specify a language for a database, you use the Database Language Setup dialog box. You display this box by choosing **Setup** from the File menu, then choosing **Language**.

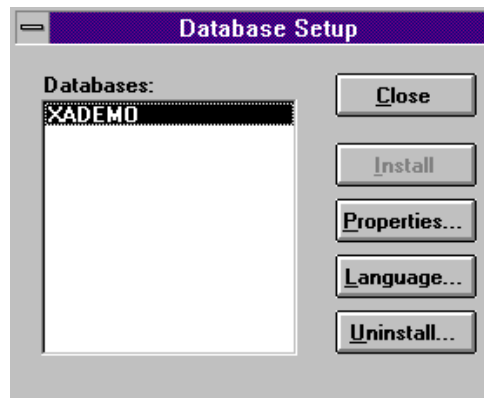


Figure 5.2 Database Setup dialog box in Administrator.

When you choose **Language**, this displays the Database Language Setup dialog box, which is identical to the System Language Setup dialog box. When you use this dialog box to add or delete languages, Administrator modifies the values in the language dimension.

Under certain conditions, you can specify a different dimension to serve as the language dimension:

- If you are just beginning to work with a database, you can specify another dimension.
- If you are working with a database that has been at least partially configured by Administrator, avoid specifying another dimension. Doing so may produce unexpected results.

Deleting a value from the language dimension has serious consequences. Several data structures used by the Data Dictionary are dimensioned by the language dimension, and information (such as level and hierarchy descriptions) is kept in those data structures. When you delete a value from the language dimension, you delete all the information for it in all relevant data structures.

The Current Language

The operating system has a current language. Each installed database can specify a current language in the Database Language Setup dialog box. The client application uses the system's current language when it displays database descriptions, selection descriptions, and other system-wide information.

The current language values can be the codes for specific languages, such as ENU for U.S. English. If you specify a current language, this value overrides the language specified in the WIN.INI file. Another value is Windows Default. If you specify this value, the client application uses the language specified in the WIN.INI file, as long as that language is an installed system language. If your users want to use the client application with the language from their own WIN.INI files, set the current language for the system and for each database to Windows Default. If the language specified in the WIN.INI file is not an installed system language, Administrator uses the first installed system language.

If you are managing more than one database for the client application, you should ensure that the languages specified in the Database Language Setup dialog box for each database are identical. In addition, these lists of

languages should be identical to that in the System Language Setup dialog box.

You can specify different current languages for different databases. The recommendation is only that the values in the Database Language Setup dialog boxes be identical; the current language specifications can be different.

Whenever you add languages, ensure that you are using the correct operating system code for that language (for example, “ENU” for English as spoken in the United States). You can check the international section of your WIN.INI file to check the operating system code.

As part of the process of installing languages, you can provide descriptive labels in the new language for dimensions, variables, formulas, and other objects. Before setting labels for a language, display the Database Language Setup dialog box and specify the appropriate language. See the section “Setting Properties of a Dimension” later in this chapter for information on labels.



For more information on languages, search for the following topic in the Administrator Help system:

“Installing Additional Languages”

Detaching a Database

When you finish working with an Express database, detach (or close) it. Be sure to detach a database before trying to attach another one with the same object names. If you attach databases with objects that have the same names, a warning message is displayed, which informs you of the conflict in object names with the database(s) that are already attached. If you choose to ignore the warning and attach the database anyway, you can create unanticipated interactions between the objects, because Administrator might load objects from the wrong database.

As you perform various operations on a database, Administrator frequently saves your changes to it. Once a database is detached, its name is removed from the Database Browser.



For information on detaching databases, search for the following topic in the Administrator Help system:

“Detaching Databases”

Ensuring that the Client Application Can Find Your Databases

You can use Administrator to configure a database so that the client application attaches it automatically. You do this by setting the `AutoAttach` property to `Yes`. If you set this property, you must ensure that the client application can find the database.

The next sections in this chapter explain how:

- The client application locates a data database when the client application attempts to attach the data database automatically.
- The client application and Administrator locate an application database. Application databases are the databases required by Administrator for configuring the data databases you create. Application databases are described briefly in the section “Databases You Should Know About” in Chapter 2.

Locating Data Databases

The data databases are the Express databases you create and to which you add data. When the client application starts, it checks the Data Dictionary to see if any data databases are configured to be attached automatically. (When you start a project or query, it also checks for any databases that are part of that project or query.) If there is such a database, the client application attempts to attach the database by searching the following locations in the indicated order:

1. The directory from which the database was attached when the project or query was last saved.
2. The directory (optionally) specified by the DBA in the database’s `Path` property when the DBA configured the database. If the client

application finds the database in the specified directory, it attaches the database and runs pre-attach and post-attach programs, if any are specified.

Locating the Application Databases

The application databases are XPDB.DB, XPDDCODE.DB, XPDDDATA.DB, and XPADMIN.DB. If Administrator or client application cannot find the application databases, it will not be able to connect to Express Server 6.0. The client application and Administrator locate XPDB.DB, XPDDCODE.DB, XPDDDATA.DB, and XPADMIN.DB by searching the following locations in the indicated order:

1. The current working directory of Express Server.
2. The paths specified in the ExpressServerDBPath setting in the server configuration on the Express Server machine. This means that all of the application databases must be located in a directory included in the ExpressServerDBPath setting, which your system administrator specifies using the Configuration Manager. You should coordinate with your system administrator to find out or request which paths are specified in the ExpressServerDBPath setting. You should also ask your system administrator for information about other server settings related to directories.

It is possible for several Express Server services to run on the same server. A service can be set up with security, so that it requires client users to identify themselves. In this case, you must edit the XADMIN.INI file so that Administrator can connect to Express Server. The same edits must also be made to the .INI file of any client application that attempts to connect to Express Server. See Chapter 2 for more information.

If an Administrator or client application user connects to Express Server using security, Administrator or the client application will first look for the location of the application databases in the path setup for the account used. If an Administrator or client application user connects to Express Server *without* using security, the default user ID is used. If that user ID has a path setup, Administrator or the client application will use this as its “home” directory.

Working with Dimensions

Dimensions organize the data that users display and analyze in the Express Client products' tables and graphs. You must define the dimensions of a database before you can add dimension values or define measures.

When you set up Express dimensions for use with the client application, you specify descriptive labels for them, and you specify whether their values should be organized according to hierarchies, and what groups of values should be saved as standard selections. The following sections describe how to define and edit dimensions, how to install them for use with the client application, and how to set their properties.



See Chapter 6 for more information on importing data into dimensions. See Chapter 4 for an overview of creating and using hierarchies, and Chapter 8 for information on saved selections.

Defining and Editing a Dimension

Once you have defined a database, you can begin to provide structure for your data by defining and editing dimensions.

Chapter 4 describes how to define dimensions and add values to them. As that chapter explains, Administrator provides dialog boxes to assist you. Refer to Chapter 4 to use the procedures for defining and editing dimensions. You can also use the Inspector to set properties for the dimension.



In addition to the information presented in Chapter 4, you can also search for the following topics in the Administrator Help system:

- “Defining Dimensions”
- “Defining Hierarchies”
- “Editing Dimensions”

Installing a Dimension

When you install a dimension, its name is automatically placed in the Data Dictionary. This step makes it possible for you to provide labels, set up hierarchies and saved selections, and specify default relations for the dimension. When you install the dimension, Administrator creates the Data Dictionary structures (Express objects) that support labels. It also installs the dimension in XPDDDATA.DB. If you later install hierarchies, Administrator creates additional structures in the database.

To see the names of Data Dictionary structures, choose **All Objects** from the View menu in Administrator. You can then find the names of Data Dictionary structures listed in the Database Browser. Some structures are also listed in the Inspector for the object that is currently selected in the Database Browser. You can recognize a structure by its name. The Data Dictionary structures for a given dimension share a common prefix, represented as *Xn* in this manual. The *X* stands for the first letter of the dimension you define. The *n* is a number that differentiates the objects for different dimensions that start with the same letter. The following are some examples of prefixes.

Dimension	Prefix for Data Dictionary Structures
PRODUCT	P0
GEOGRAPHY	G0
PATTERN	P1

To take advantage of the configuration tasks you perform in Administrator, your client code can refer to the Data Dictionary structures Administrator generates.



For more information on installing a dimension, search for the following topic in the Administrator Help system:

“Installing Dimensions”

Setting Properties of a Dimension

You can specify the way the client application will handle a given dimension by specifying its properties. The properties of a dimension are a group of its characteristics, such as the dimension's description and labels. The values that you set as you define and edit dimensions in Administrator are stored in the properties. You can also set properties in the Inspector.

Table 5.2 lists the dimension properties available on the Properties tab of the Inspector, and Table 5.3 lists the properties available on the Advanced tab. The advanced properties measure the efficiency of dimensions and variables. You can use the information specified by these properties to improve the response time of very large dimensions and variables and to decide whether to optimize them. Not all properties are visible if the dimension is not installed.

The name expression properties (DescriptionExp, DescriptionPluralExp, ShortNameExp, and ShortNamePluralExp) are used when you have more than one name for the same dimension value. For example, you might want to set the name expression properties if your database includes more than one language, so that you can provide a different set of dimension names for each language.

Table 5.2 Dimension Properties

Property	Description
# Dimension Values	The number of values in the dimension (read-only).
Database	The name of the database to which the dimension belongs (read-only).
DataType	The data type of the dimension, such as TEXT (read-only).
Description	A singular, long form of the dimension name used in the Selector, a tool in Express client applications. You can specify a plural description using the DescriptionPlural property.
DescriptionExp	An Express expression that evaluates to a text value. When you enter this expression, the client application evaluates it and places the result in the Description property.
DescriptionPlural	A plural, long form of the dimension name used in the Selector.
DescriptionPluralExp	An Express expression that evaluates to a text value. When you enter this expression, the client application evaluates it and uses the result to display plural descriptions.

Table 5.2 Dimension Properties

Property	Description
Dimensioned By	For conjoint dimensions, the names of the other base dimensions (read-only).
DimensionType	Indicates whether the dimension has a time data type. The value 1 indicates the dimension has a time data type; the value 0 indicates that it does not.
ID Code	An identification code that Administrator assigns to the dimension (read-only).
Installed	A boolean expression indicating whether the dimension is installed for use with the client application (read-only).
IsAttribute	A boolean expression indicating whether a conjoint dimension is treated as an attribute in the Selector.
LD	The long description for the dimension.
LongLabelVar	The name of the variable that holds the long labels for the dimension's values. The default name is <code>XN.LONGLABEL</code> .
MDBSource	The name of an Express Server counterpart to an object in another remote Express database. Used for remote data.
ObjectType	The object's type — in this case, dimension (read-only).
ShortLabelVar	The name of the variable that holds the short labels for the dimension's values. The default name is <code>XN.SHORTLABEL</code> .
ShortName	A singular, short form of the dimension name used in the Express Client products' tables, graphs, and the Selector if there is no description.
ShortNameExp	An Express expression that evaluates to a text value. When you enter this expression, the client application evaluates it and places the result in the ShortName property.
ShortNamePlural	A plural, short form of the dimension name used in the Express Client products' tables, graphs, and the Selector if there is no plural description.
ShortNamePluralExp	An Express expression that evaluates to a text value. When you enter this expression, the client application evaluates it and uses the result to display short plural names.
UserVisible	Whether to make this dimension available to client application users. When you set this property, it will take effect only in the Express Client products.
Width	The width of the object. Applies only to text dimensions (read-only).

Table 5.3 lists the advanced object properties. These properties are read-only. See the OBJ function in the Express Language Help for more information on these properties.

Table 5.3 Advanced Object Properties (Read Only)

Property	Description
Disk Size (bytes)	The total number of bytes needed to store the object.
In Order	A boolean expression indicating whether the logical order of the values of an Express object's underlying dimensions matches the physical order in the database file. TRUE indicates that the logical order matches the physical order.
Number Active Cells	The number of active value cells.
Number Deleted Cells	The number of deleted value cells.
Number Segments	The number of segments associated with any object that has more than one dimension. Express uses segments internally to keep track of the physical storage of the object's values.



For information on setting properties, search for the following topic in the Administrator Help system:

“Setting Object Properties”

Working with Measures

Measures are stored in your Express database as formulas, relations, and variables.

A *formula* is an expression that you name and define.

A *relation* describes a correspondence between the values of two or more dimensions. A relation can have more than one dimension, but the values of the relation must be values from the related dimension.

A *variable* is an array whose cells hold individual data values. For more information about measures, see Chapter 1.

When you set up measures with Administrator, you specify whether the client application should give users access to the measures, how the

measure names will be displayed, and, for numeric measures, how their data will be formatted. Whether a given measure is stored as a formula, as a relation, or as a variable makes no difference in the steps you take in configuring it for the client application.

The following sections describe how to work with measures:

- Defining and editing measures
- Installing measures for use with the client application
- Managing data-driven formatting
- Setting properties for measures
- Setting up for SQL reach-through
- Setting number formats
- Displaying descriptions for dimension values



See Chapter 6 for information on importing data into measures. See Chapter 4 for an overview of creating and using hierarchies, and Chapter 8 for information on specifying default relations.

Defining and Editing a Measure

After defining and editing dimensions, you continue providing structure for your data by defining and editing formulas, relations, and variables.

Chapter 4 describes how to define a variable and add values to it. As that chapter explains, Administrator provides dialog boxes to assist you. Refer to Chapter 4 to use the procedures for defining and editing variables. The procedures for defining and editing other measures (formulas and relations) are very similar and are described in detail in the Administrator Help system, as specified later in this section. You can also use the Inspector to set properties for a measure.

While editing a measure with six or fewer dimensions in the Edit dialog boxes, you can rotate the data display by exchanging the position of the dimensions of the measure. For example, if the display shows time periods down the first column and line items on separate pages, you can rotate them

so that line items appear down the first column and time periods appear on separate pages. Simply click on the dimension name and drag it onto another dimension name. The dimensions will exchange positions.

The Express Client products display the data for your measures in tables and graphs. In addition, the user sees measure names when selecting and aggregating data. The client application has default methods for displaying measures. However, you can customize how numeric data in installed measures appears to the user. To do so, you specify number formatting information when you define and edit measures. You can specify information such as how many digits to display for decimal numbers, how positive and negative numbers appear, and what symbol to use for currency.

The Data Dictionary stores these specifications in the *number format string* for your measure. All the number format information is stored in this single string so that the client application will not have to check the values of several properties each time it displays data. When you change a measure's number format properties, its number format string is automatically recalculated. In tables, client application users can override the formatting you provide for a measure.

If you do not specify particular values for a measure's number format properties, the Data Dictionary uses the default number formats for your system. The defaults are described in the section "Setting Number Formats" later in this chapter.

Because the Excel Add-In displays data in spreadsheets, any number formatting information you specify in Administrator will not have an effect in Excel.



For information on defining and editing measures, search for the following topics in the Administrator Help system:

- "Defining Formulas"
- "Defining Relations"
- "Defining Variables"
- "Editing Formulas"
- "Editing Relations"
- "Editing Variables"

Installing a Measure

When you install a measure, its name is automatically placed in the Data Dictionary. This step makes it possible for you to specify how the measure will look in the client application. You should install all the measures to which the client application user will have access.

In addition, if your Express Client product users will perform table write-back during what-if analysis, you must do the following:

- Install variables to use for this purpose.
- Ensure that the databases containing the variables are attached as read-only, so that the changes made during table write-back do not become permanent.

If the client application is the Excel Add-In, your users will not be able to perform write-backs.

In the Express Client products, the table property that enables table write-back must be set. In addition, you should define and install formulas that use the variables installed for write-back. If you do not install formulas that the variables will affect, client application users must create custom measures for what-if analysis.



For more information on installing measures, search for the following topic in the Administrator Help system:

“Installing Measures”

Managing Data-Driven Formatting

In the Express Client products, you can create tables that contain data from formulas, relations, and variables. You can specify that the data in each cell of a table be formatted in a certain way, based on the value of the data.

To make these specifications, you create a data-driven formatting formula for an installed measure. The formula must have the same dimensions or a subset of the dimensions of the measure for which it specifies formatting.

Users can see the formula names and explanations in the Data-Driven Formatting dialog box.

You can easily create data-driven formatting formulas, and you can even create multiple formulas that can be shared across measures with the same dimensionality. You can specify one formula as the default one, but client application users can choose any formula you provide.

If the client application is the Excel Add-In, any data-driven formatting formula you create in Administrator will not have an effect in Excel spreadsheets.



For more information on data-driven formatting formulas, search for the following topics in the Administrator Help system:

“Defining a Data-Driven Formatting Formula”

“Entering Code for a Data-Driven Formatting Formula”

“Installing Measures”

Setting Properties of a Measure

You can specify the way the client application will handle a given measure by specifying its properties. The properties of a measure include characteristics such as the measure’s description and labels. The values that you set as you define and edit measures in Administrator are stored in the properties. You can also set properties in the Inspector.

Table 5.4 lists the properties for measures. Formulas and relations have no advanced properties. Variables have the same advanced properties that dimensions have, with one addition: variables have the NA Pages property, which shows how many pages of NA information the variable has.

Table 5.3 describes the advanced properties of dimensions and variables. Not all properties are visible if the measure is not installed.

Table 5.4 Measure Properties

Property	Description
Compiled	Information about the compilation status of a formula (read -only).
Database	The name of the database to which the measure belongs (read-only).
DataType	The data type of the measure, such as TEXT (read-only).
Description	A long form of the measure name used in the client application.
DescriptionExp	An Express expression that evaluates to a text value. When you enter this expression as a property, Administrator evaluates it and places the result in the Description property.
Dimensioned By	The names of the dimensions by which the measure is dimensioned (read-only).
FormatFormula	Identifies the default data-driven formatting formula specified for an installed variable, relation, or formula (read -only).
Installed	A boolean expression indicating whether the measure is installed for use with the client application (read-only).
IsAttribute	A boolean expression indicating whether a relation is treated as an attribute in the Selector.
LD	The long description for the measure.
MDBSource	The name of an Express Server counterpart to an object in another remote Express database.
NA Pages	For variables only, identifies the number of pages of disk space that contain only NA values (read -only).
ObjectType	The object's type — in this case, formula, relation, or variable (read-only).
ShortName	A short form of the measure name used in the Express Client products' tables, graphs, and the Selector if there is no description.
ShortNameExp	An Express expression that evaluates to a text value. When you enter this expression as a property, Administrator evaluates it and places the result in the ShortName property.
StorageType	The storage type of the variable, either Temporary, In-Place, or Permanent (indicated by a blank setting). (Read-only.)
UserVisible	Whether to make this measure available to client application users.
Width	The width of the object. Applies only to dimensioned text, integer, and boolean variables (read-only).

Name Properties

The name properties specify the words to be used when the client application displays the measure name. You can specify a short name and a description, and name expressions for the measure in the following properties:

Description
DescriptionExp
ShortName
ShortNameExp

If you are satisfied with specifying a single description and short name, you can leave the DescriptionExp and ShortNameExp properties blank. However, if your database includes more than one language, you might want to use the name expressions to provide a different set of measure names for each language.



For more information on descriptions, search for the following topics in the Administrator Help system:

“Installing Additional Languages”
“Setting Object Properties”

Setting Up for SQL ReachThrough

If you have Express Server set up for SQL support, you can connect to SQL databases. These connections provide dynamic SQL reach-through that enables you to set up Express formulas that fetch data from a relational database as client application users request the data.

You can maintain several SQL connections, but only one can be activated at a time. After making a connection, you use a formula to map the data between the SQL database and Express. As part of mapping the data, you can customize the SQL data fetching by preparing a selection preprocessor script.

When client application users work with the formulas that fetch SQL data, they will use these connections. If users need user names and passwords to

access the SQL databases, you should either provide the names and passwords or prompt for them in the pre-attach processor program of the databases that contain the formulas.

See Chapter 10 for details about setting up a system for SQL connections. It also describes how to prepare the pre-attach processor program for SQL connections.



For more information on SQL reach-through, search for the following topics in the Administrator Help system:

“Connecting to SQL”

“Mapping SQL Data”

“Preparing a Selection Preprocessor Script”

Setting Number Formats

The number format specifications you make when you define or edit a measure apply to that one measure. For example, if you change the number of decimal places from two to three, the change affects only the measure that you are editing. If you want to make such a change for all the measures that the client application displays, you can change the default setting for the system.

Note: If the client application is the Excel Add-In, any number formatting you specify in Administrator will not take effect in Excel spreadsheets.

The following sections explain how to set the default number formatting for the system and how to specify that a certain measure uses those defaults.

Setting the Defaults

When calculating the number format string for a given measure, Administrator uses the current number format specifications. If you do not specify a value, the corresponding default value is used. Thus, you can change a number format characteristic for all measures in the system by changing the default setting and ensuring that the corresponding setting is blank for all measures.

Because the number format string for a given measure is stored in the measure's database, only the strings for measures in attached databases can be recalculated. Therefore, to make a system-wide change, attach all possible databases before changing the default values. If there are two databases with conflicting definitions (e.g., two measures named SALES), attach only one of the databases before changing the default values. Later, you can attach the other database and explicitly recalculate its number format strings.

Notes: You cannot set system defaults for the currency symbol, decimal separator, and thousands separator, although you can set these for individual measures. The client application inherits these from the settings of *sCurrency*, *sThousand*, and *sDecimal* in the [intl] section of the WIN.INI file.

The Express Client product users can also set the format of numbers. If you set number formatting in Administrator, users can override the formatting in tables but not in graphs. For information on the precedence of number formatting in the client application, see the user's guide for the application.

The following procedure describes how to set number format defaults. The procedure suggests creating a "dummy" variable to use as the model for the defaults. You can set the number formatting for this variable the way you would like the defaults to be.



To set a number format default:

1. Attach the databases that have installed measures, but that do not have conflicts in definitions (e.g., objects with the same name). In Administrator, you can attach only four databases at one time.
2. In one of the databases, create a variable with a numeric type, such as DECIMAL.
3. Select the Format tab.
4. Set the values as you want the defaults to appear.
5. When the settings are correct, choose **Set Default**. Administrator automatically calculates the number format string for all the measures in all the attached databases.

6. Choose **Close**. Choose **No** when you are prompted to decide whether to finish defining before closing.
7. If you have not attached some databases because they have conflicting definitions, update their defaults with the following steps:
 - a. Detach the currently attached databases that have conflicts.
 - b. Attach the databases that need to have their defaults updated.
 - c. Repeat Steps 2 through 6.

Using the Defaults

If you have edited the number formatting for a particular measure, then decide that you want to remove your edits and use the default settings, use the following procedure.



To have a particular measure use the default number format settings:

1. Display the measure in the appropriate Edit dialog box.
2. Select the Format tab.
3. Choose **Use Defaults**.

Displaying Descriptions for Dimension Values

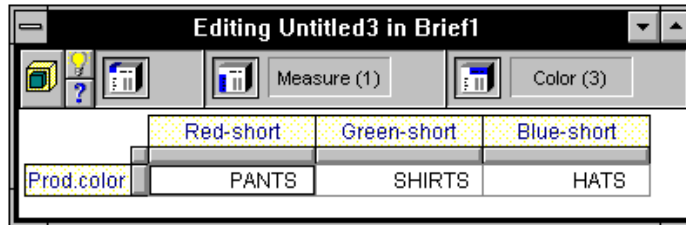
In a database, you might have a relation that contains data for your client application users to view. If users can create a table for that relation (Excel Add-In users cannot create tables), they will see the dimension values displayed exactly as they are stored in the dimension. You might want to display the values in a different way, such as by showing their long labels.

To do so, you create a formula that displays the values in the desired form. You make the formula user-visible, and you hide the original relation. The following simple example illustrates this process.

Suppose you have a dimension called **PRODUCT**, whose values are **PANTS**, **SHIRTS**, and **HATS**. You have a dimension called **COLOR**, whose values are **RED**, **GREEN**, and **BLUE**. You also have a relation called **PROD_COLOR** between the two dimensions. In the relation,

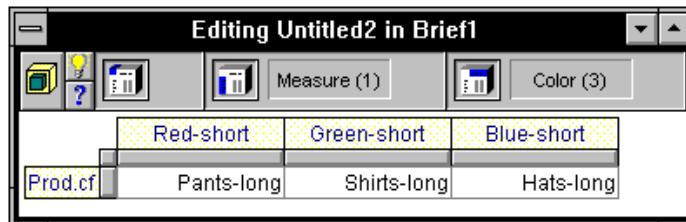
PRODUCT is the related dimension (or data type) of the relation and COLOR is the dimension of the relation.

In Administrator, if you installed PROD_COLOR and specified it as user-visible, an Express Client product user could table the relation and see a table such as the following.



	Red-short	Green-short	Blue-short
Prod.color			
	PANTS	SHIRTS	HATS

However, you might prefer that users see the long label for each color, instead of the actual dimension values, as shown in the next figure.



	Red-short	Green-short	Blue-short
Prod.cf			
	Pants-long	Shirts-long	Hats-long

You must create a formula that displays the long labels, so Express Client product users can create a table with that formula. The following procedure describes this process.



To set up a formula in Administrator to display the long labels:

1. Install the COLOR dimension and specify it as user-visible.
2. Install the PRODUCT dimension and specify it as user-visible.
3. Do not install the PROD_COLOR relation and do not specify it as user-visible.
4. Define a text formula called PROD_CF and dimension it by COLOR.
5. Install PROD_CF and specify it as user-visible.
6. In the equation for PROD_CF, include the name of the formula that stores the long labels for the values of the PRODUCT dimension. For

example, if the formula for the long labels is called P3.LONGLABELF, the equation for the PROD_CF formula would be the following.

```
p3.longlabelf(product prod_color)
```

Executing Express Commands

You can execute Express commands from within Administrator by using the Express Command dialog box, as shown in Figure 5.3. You display this box by choosing **Express Command** from the Tools menu.

This dialog box contains two boxes. In the Type Express Commands box, you enter the commands that you want Express to execute. You can enter each command on its own line. In the Command Result box, Express displays the output from the commands.

Note: You can execute most Express commands from Administrator. Do not use local Express commands such as HELP, LOG, PAUSE, SESSION, TRACE, or WATCH. Do not use the Express OUTFILE command, which does not work from the Express Command dialog box. Do not use commands that require user input. In addition, do not attach or detach databases or perform operations on objects such as defining, copying, deleting, and renaming; your actions will not be reflected in the Database Browser. Also, you will be defeating the purpose of Administrator's ease of use. Instead, use Administrator's menus and dialog boxes to perform these tasks.

Use the Express Command box to execute commands such as LIMIT. These commands are helpful when you perform tasks such as executing programs and editing dimensions, measures, and other objects. See the Express Language Help for information about using the LIMIT command. Chapter 9 also gives a brief discussion of some common Express commands you may use, including the LIMIT command.

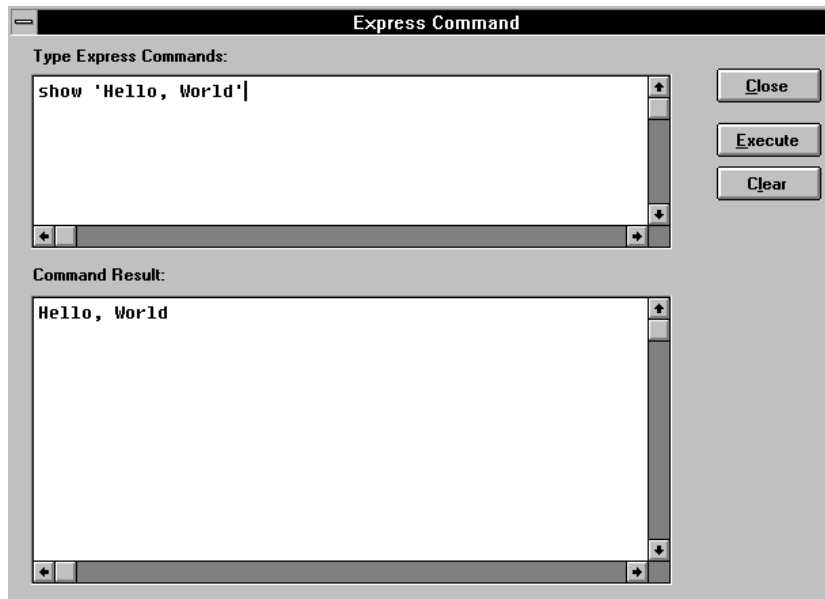


Figure 5.3 Express Command dialog box.



For more information on executing Express commands, search for the following topics in the Administrator Help system:

- “Executing Express Commands”
- “Express Command Dialog Box”

Chapter 6

Importing and Exporting Data

Once you have created a database and defined the dimensions and measures that provide structure for your data, you can read the data into the database from a file.

This chapter describes the following:

- Importing data from various sources
- Exporting data and definitions to Express Interchange Format (EIF) files
- Summary

You can import data from structured or ruled ASCII files (.PRN or .TXT files) or from relational database (SQL) files. (You can also connect to SQL databases directly without importing data, as described in Chapter 10.) You can import data and definitions from other Express databases using EIF files.

You can export data and definitions to an EIF file only.

Importing Data

When you import data from an EIF file, you can import the database object definitions at the same time. This allows you to import the data and its structure simultaneously.

Before importing data from any other source, you must know the structure of the data in the source file. The data structure determines the dimensions,

variables, relations, and other Express objects you need to store the data properly in the Express database.

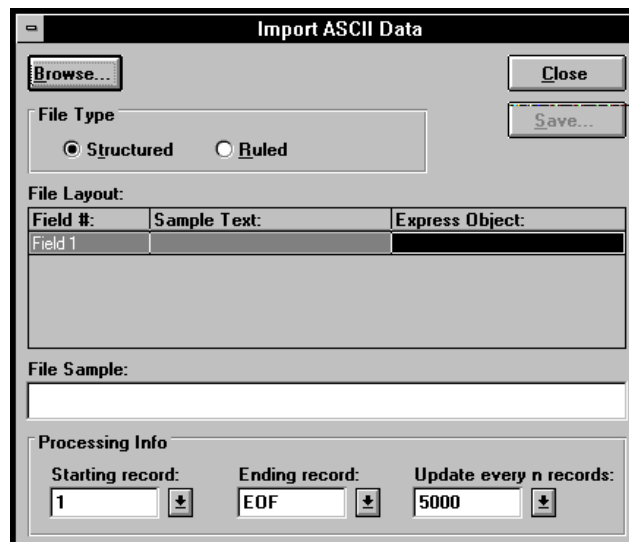
The structure of the data can also affect the import process when you import data from an ASCII file. For example, if an ASCII file contains values that will be stored in a relation, and it also contains the values of one of the relation's base dimensions, you must import the base dimension values before you import the relation values.

Importing Data from an ASCII File

You can import data from ASCII files. The files can be *structured*, in which text fields are identified by quotes and numbers are identified by spaces or commas. You can also import *ruled* ASCII files, where each field occupies a certain number of columns in the file (one column being one character or space).

When you import ASCII data, you follow these basic steps:

1. Create Express objects to store the data, as described in Chapter 4 and the Administrator Help system.
2. From the Tools menu, choose **Import ASCII** to display the Import ASCII Data dialog box.



3. Create a data loader program to read in the data. The Import ASCII Data dialog box makes this easy by prompting you to specify the necessary information, such as:

- The file from which you want to import data
- Information about the structure of the data in the source file
- The Express object that will store each field in a data record
- The first and last data record to read into the database

After you provide the information, Administrator creates a data loader program.

4. Execute the data loader program. You can do this as you exit the dialog box, or you can execute it separately. The program appears under Programs in the Database Browser. You can also edit the program.

You do not have to load all the fields in a source file with the same data loader program. You can read the same file several times, reading in different information each time.

You might need to use more than one data loader program to import complex source files. For example, one file might contain all the information for a dimension, including hierarchical information stored in a relation. Before you can import the relation values, you must import all the values for the base dimensions. In this case, you would create one data loader program to import the base dimension values and another to import the relation values. You can call both programs in the Express Command dialog box and execute them. For more information about importing ASCII data and data reader programs, see the *Express Language Programming Guide*. Instead of using the Import ASCII Data dialog box, you can enter Express commands in the Express Command dialog box to import ASCII data. Refer to the IMPORT (Text File) command in the Express Language Help system.



To see detailed instructions for importing ASCII data, search for the following topics in the Administrator Help system:

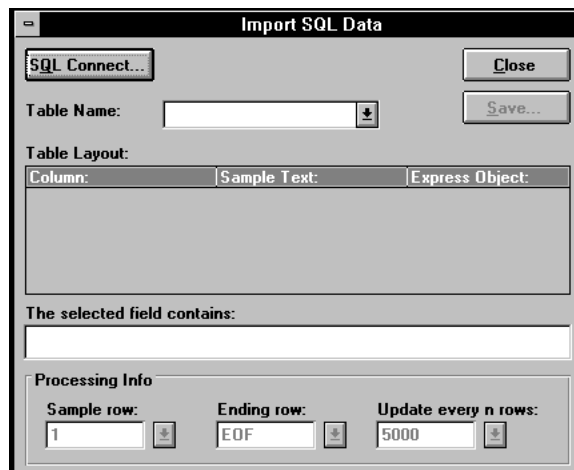
“Import ASCII Data Dialog Box”
“Importing ASCII Data”

Importing Data from a Relational Database

You can import data from a SQL data source into an Express database. Before you connect to Express Server, make sure your system administrator has installed the appropriate SQL support.

When you import data from a relational database, you follow these basic steps:

1. Create Express objects to store the data, as described in Chapter 4 and the Administrator Help system.
2. From the Tools menu, choose **Import SQL** to display the Import SQL Data dialog box.



3. Create a data loader program to read in the data. The Import SQL Data dialog box makes this easy by prompting you to specify the necessary information, such as:
 - The data source from which to import data
 - The SQL table from which to import data
 - The Express object that will store each field in a data row
 - The last data row to read into the database

After you provide the information, Administrator creates a data loader program.

4. Execute the data loader program. You can do this as you exit the dialog box, or you can execute it separately. The program appears under Programs in the Database Browser. You can also edit the program.

You do not have to load all the fields in a data source with the same data loader program. You can read the same data source several times, reading in different information each time.

You might need to use more than one data loader program to import from a complex data source. For example, one table might contain all the information for a dimension, including hierarchical information stored in a relation. Before you can import the relation values, you must import all the values for the base dimensions. In this case, you would create one data loader program to import the base dimension values and another to import the relation values. You can call both programs in the Express Command dialog box and execute them.



To find detailed instructions for importing data from a relational database, search for the following topics in the Administrator Help system:

- “Import SQL Data Dialog Box”
- “Importing Data from Relational Databases”

Importing Data from an EIF File

You can easily import data from an Express database, in the Express Interchange Format (EIF). You do not have to create the Express database objects to store the information, as you do with data from other sources. When you import the data, you import the definitions as well.

When you import EIF data and definitions into a database, you follow these basic steps:

1. In the Database Browser, select the target database, which will hold the imported data.
2. From the Tools menu, choose **Import EIF** to display the Import an EIF File dialog box.
3. Specify the file from which you want to read data and definitions.

4. Select the source database objects that you want to import. You do not have to import all objects in one import operation.

You can also import only the definitions in one import operation and import the data after you import the definitions.

The object definitions in the target database must match the definitions in the source database from which the EIF was created, in both name and data type. Each time you import EIF data, you can append new dimension values, or you can choose to read in only values that match the dimension values already in the target database.

Instead of using the Import an EIF File dialog box, you can enter Express commands in the Express Command dialog box to import EIF files. Refer to the IMPORT (EIF File) command in the Express Language Help system.



To see complete directions for importing EIF data, search for the following topics in the Administrator Help system:

“Import an EIF File Dialog Box”

“Importing EIF Files”

Note: After importing EIF data, configure the database as described in Chapter 8.

Exporting Data to an EIF File

You can export the data from a source database to an EIF file. Once the data is in the EIF file, you can import the data into a target database.

Exporting data to an EIF file allows you to copy data and object definitions from one Express database to another.

Note: The status of the database objects determines which values are exported to the EIF file. Execute the Express STATUS command in the Express Command dialog box to see the current status of each object. Then use LIMIT commands or ALLSTAT to set the status before you export the data. See Chapter 9 for more information about using these commands.

When you export data to an EIF file, you follow these steps:

1. In the Database Browser, select the database that contains the data you want to export.
2. Execute the LIMIT command to make sure you get the data you want.
3. From the Tools menu, choose **Export EIF** to display the Export an EIF File dialog box.
4. Specify a file name for the exported data.
5. Specify the objects you want to export.

Instead of using the Export an EIF File dialog box, you can enter Express commands in the Express Command dialog box to export EIF files. Refer to the EXPORT (EIF File) command in the Express Language Help system.



To find detailed instructions for exporting data to an EIF file, search for the following topics in the Administrator Help system:

- “Export an EIF File Dialog Box”
- “Exporting to EIF Files”

Chapter 7

Manipulating Data

After you have created databases and their dimensions and measures and have put data in them, you can use other Express objects to manipulate that data. This chapter explains the following:

- Creating valuesets
- Creating programs and models
- Copying, renaming, and deleting objects
- Setting properties
- Searching for objects
- Summary

Note: The first two sections of this chapter describe how to work with valuesets, programs, and models in Administrator. For more information on these Express objects, see Chapter 1.

Creating Valuesets

A *valueset* is an Express object that contains a list of dimension values for a particular dimension. First, you define a valueset. You then use the LIMIT command in the Express Command dialog box to assign values from the dimension to the valueset. Once you have created a valueset, you can use it with Express commands in programs. For example, you can use the LIMIT command to limit the status of a dimension to a valueset.

See the Express Language Help for information about using the LIMIT command. Chapter 9 also gives a brief discussion of some common Express commands you may use, including the LIMIT command.



To define a valueset in Administrator:

1. From the Dictionary menu, choose **Define**, then choose **Valueset** to display the Define a Valueset dialog box.
2. Specify the appropriate settings in the dialog box.
3. Choose **Define**, then **Close**.

When you first define a valueset, its value is NULL. You must then assign dimension values to the valueset.



To assign dimension values to a valueset:

1. From the Tools menu, choose **Express Command** to display the Express Command dialog box.
2. In the dialog box, use LIMIT commands to set the status of the valueset's dimension. The dimension values in the status are automatically assigned to the valueset. For example, suppose you want to limit the MONTH dimension to the even-numbered months of 1996. After creating a valueset named EVEN96, you would assign dimension values to the valueset with the following commands.

```
define even96 valueset month
limit even96 to 'FEB96' 'APR96' 'JUN96' 'AUG96' -
'OCT96' 'DEC96'
```

3. When you have finished editing, choose **Close**.

You must choose **All Objects** from the View menu in Administrator to see valuesets listed in the Database Browser.



For information on defining valuesets, search for the following topic in the Administrator Help system:

“Defining Valuesets”

Creating Programs

A *program* is a stored procedure that acts on Express data or external data. A program, like a dimension or a variable, is a database object. The data loader programs described in Chapter 6 are examples.

A *model* is an Express object that contains a set of interrelated equations. The calculations in an equation can be based either on variables or on dimension values. Chapter 1 describes programs and models in more detail.

You can use Administrator to define, edit, compile, and execute programs and models.

The following sections describe the steps you take to create a program. The steps for creating a model are similar and are described in detail in the Administrator Help system.

Defining a Program

The first step in creating a program is to define the program object. Then you can enter your code by editing the program. Refer to the DEFINE PROGRAM command in the Express Language Help system for more information about defining programs.



To define a program:

1. From the Dictionary menu, choose **Define**. Then choose **Program** to display the Define a Program dialog box.
2. In the Name box, type in a valid Express object name for your program.
3. In the LD box on the Label tab, type in a long description for the program. It is not displayed to client application users.
4. In the Data Type box on the General tab, specify the data type of the return value of the program. If the program will not have a return value, use the default type of *None*.
5. In the Database box, select the database where you want to create the program.

6. Choose **Define** to create the program object definition. The name of the program will appear in the Database Browser under the heading Programs. The program object does not yet have any code associated with it. To enter the code, you must edit the program.
7. Choose **Close** to return to the Main Window.

The next section describes how to edit the program object you have just created.

Editing a Program

After defining an Express program object, you can attach the programming code by editing the object. You edit programs using the Edit Window within Administrator. When you use this window, the menu bar in the Main Window changes to have three menus — File, Edit, and Tools.



To edit a program:

1. Select the program in the Database Browser, then choose **Edit** from the Dictionary menu.
2. Enter the programming code in the Edit Window. You can use standard Windows editing techniques to cut, copy, and paste text, using commands from the Edit menu. If you wish, you can save your changes periodically to the Express database by choosing **Save** from the File menu.

Once you have edited your program, you can compile it without leaving the editing environment. Refer to the *Express Language Programming Guide* for information about program contents.

Compiling a Program

With the Edit Window still open, you can easily compile your program.



To compile a program, from the Tools menu, choose **Compile**.

The Compile command saves and compiles the code in the Edit Window.

Fixing Compilation Errors

If your program does not compile successfully in the Edit Window, you see error messages in the Compilation Errors dialog box. You can then fix compilation errors.



To fix compilation errors:

1. Modify your code in the Edit Window.
2. When you are ready to recompile your changes, choose **Compile** again from the Tools menu.

Once you have fixed any compilation errors, you can execute your program.

Executing a Program

Once you have successfully compiled your program, you can then execute your program from the Edit Window.



To execute a program:

1. If you are not already in the Administrator editing environment, select your program in the Database Browser, then choose **Edit** from the Dictionary menu.
2. From the Tools menu, choose **Execute** (or from the popup menu you display with the right mouse button). The Express Command dialog box is displayed.
3. Choose **Execute**. Your program is executed. If your program produces output, that output is displayed in the Express Command dialog box.
4. Choose **Close** to exit the Express Command dialog box.
5. When you are through working with your program in the Edit Window, choose **Close** from the File menu.

If your program contains errors that require investigation, such as stepping through code, there are a couple of alternatives for debugging your program.

Debugging a Program

If you find that you need more powerful debugging tools (e.g., so that you can step through your program), you can debug your program in one of the following ways.

In the Express Command window, you can set the PRGTRACE and ECHOPROMPT options to YES, then execute the DBGOUTFILE command. This creates a log of your program's execution, including error messages. Outside of Administrator, you can examine the debugging file with a text editor. You can then edit your program and compile it again in Administrator. For more information, see the DBGOUTFILE command in the Express Language Help system and the debugging section in the *Express Language Programming Guide*.

Alternatively, if you have Personal Express 5.x (a separate product that is available from Oracle), you may use the Express debugger through an XCA connection. In this case, you may define and edit your program in the Personal Express 5.x editor, define an identical program in your Express database, then use EXPORT PIPELINE to send the program to your database in Express 6.0 or later.



See the *Express Language Programming Guide* for more information about writing and debugging programs.



For information on defining, editing, compiling, and executing programs and models, search for the following topics in the Administrator Help system:

- “Defining Models”
- “Defining Programs”
- “Edit Menu Bar”
- “Edit Window”
- “Editing Programs and Models”
- “Executing Programs and Models”

Copying, Renaming, and Deleting Objects

After defining objects in a database, you can copy, rename, and delete them, as described in the following sections.

Copying Objects

Once you have defined an object, you might want to define another object that is similar to the first, but with minor differences. You can copy the first object and use it as the basis for the second one. You can then edit the second object to include the appropriate differences.

If you copy an installed object, the new object created by the copy is not installed. The properties for the new object that relate to installation are blank; they do not retain the property settings of the original object. Therefore, you need to set the new object's properties just as if you had defined a new object.



For more information on copying objects, search for the following topic in the Administrator Help system:

“Copying Database Objects”

Renaming Objects

After defining an object, you might want to give the object another name. You can change the name of uninstalled variables, dimensions, and so forth. However, you cannot rename objects created by Administrator to support installed and hierarchical dimensions. These database objects have a prefix of Xn , such as G1.ORDER. The X stands for the first letter of the dimension you define. The n is a number that differentiates the objects for multiple-user dimensions that start with the same letter.

You cannot rename an installed object. You must first uninstall it, which deletes its supporting objects. You can then rename and reinstall the object. See Chapter 8 for information on uninstalling objects.

If you rename an object, you must manually change any references to it that appear in other objects such as programs, models, and formulas.



For more information on renaming objects, search for the following topic in the Administrator Help system:

“Renaming Database Objects”

Deleting Objects

After defining an object, you might find that you no longer need it and want to delete it. You can delete uninstalled variables, dimensions, and so forth. However, you cannot delete objects created by Administrator to support installed and hierarchical dimensions. These database objects have a prefix of *Xn*, such as G1.ORDER.

You cannot delete an installed object. You must first uninstall it, which deletes its supporting objects. See Chapter 8 for information on uninstalling objects.

You cannot delete a dimension that is used to define a variable, formula, or other object. You must first delete the object dimensioned by the dimension. You must manually edit any references to the deleted object in programs, models, and so forth.



For more information on deleting objects, search for the following topic in the Administrator Help system:

“Deleting Database Objects”

Setting Properties

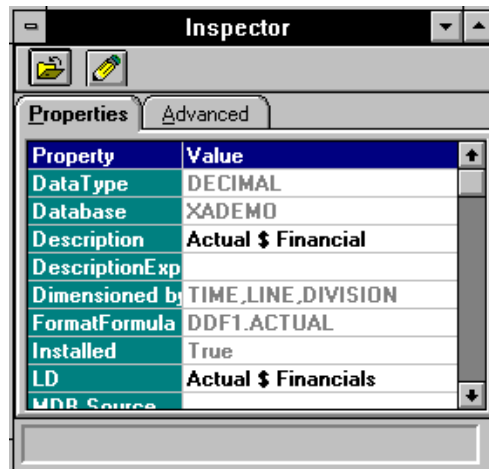
The objects that you work with in Administrator have *properties*. A property specifies an attribute of an object. For example, the properties for a dimension specify characteristics, such as its description and whether it is visible to the client application user.

You set properties for objects when you edit the objects, as described in the Administrator Help system. You can also set properties in the Inspector for all objects except databases.



To set a property for an object in the Inspector:

1. In the Database Browser, select the object you want to inspect.
2. From the Window menu, choose **Inspector** to display the Inspector.



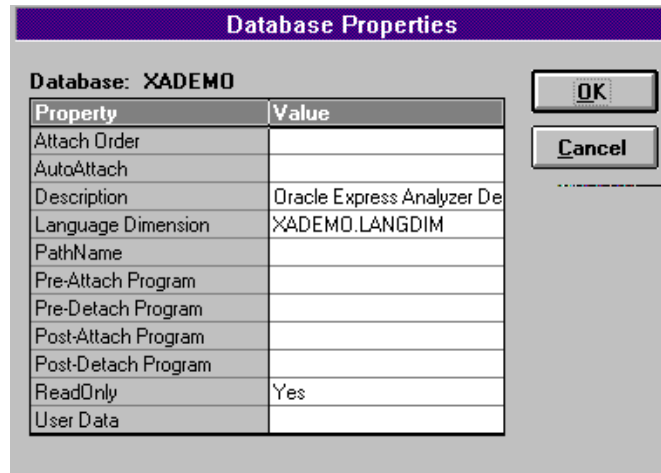
3. Click on the value in the Value column for the property you want to set. This selects the current value. You can double-click to enter edit mode.

Note: Dimmed properties are read-only.

4. Type in a new value.
5. Press ENTER.

**To set a property for a database:**

1. From the File menu, choose **Setup** to display the Database Setup dialog box.
2. In the dialog box, choose **Properties** to display the Database Properties dialog box.



Property	Value
Attach Order	
AutoAttach	
Description	Oracle Express Analyzer De
Language Dimension	XADEMO.LANGDIM
PathName	
Pre-Attach Program	
Pre-Detach Program	
Post-Attach Program	
Post-Detach Program	
ReadOnly	Yes
User Data	

Note: If the **Properties** button is dimmed, choose **Install** to install the database. Then choose **Properties**. You can set properties only for installed databases.

3. Click on the value in the Value column for the property you want to set. This selects the current value. You can double-click to enter edit mode.
4. Type in a new value.
5. Press ENTER.



For more information about setting properties, search for the following topics in the Administrator Help system:

- “Editing Dimensions”
- “Editing Formulas”
- “Editing Programs and Models”
- “Editing Relations”
- “Editing Variables”

“Installing Express Databases”

“Setting Object Properties”

Searching for Objects

If you are working with one or more databases that contain many objects, you might lose track of particular objects. If this happens, you can search for the objects in one or more attached databases. You can search using various criteria, including the following:

- Whether the object is dimensioned by a certain object
- What the object’s type is (dimension, variable, etc.)
- What the object’s data type is (Decimal, Text, etc.)
- Whether the object’s name contains certain characters
- Whether the object is referenced by another object
- Whether the object is visible to the user

Once you specify the criteria, the results of your search are displayed in the Database Browser. Searching for user-visible objects is especially helpful when viewing lists of dimensions and measures with which client application users will work.

You can switch between viewing all objects in the Database Browser, viewing user-visible objects, and viewing the results of your search by choosing options from the View menu. The following operations affect the display of the Database Browser:

- Whenever you choose **Search Results**, Administrator executes a search using the most recently specified search criteria.
- Whenever you perform a database operation (such as attaching or detaching database files), Administrator disables the **Search Results** menu item, and you must define the search criteria again.



For more information on searching for objects, search for the following topics in the Administrator Help system:

“Search for Database Objects Dialog Box”

“Searching for Database Objects”

Chapter 8

Configuring Databases for a Client Application

This chapter explains how to use Administrator to configure a database for use with a client application. The first section contains a configuration check list, and the following sections describe the parts of the configuration process.

- A configuration checklist
- Installing databases, dimensions, and measures
- Hiding dimensions and measures
- Uninstalling databases, dimensions, and measures
- Editing dimension hierarchies
- Defining a dimension's selections
- Specifying default relations
- Defining system shutdown programs
- A Simple Case Study
- Summary

When you use Administrator to configure databases, it places configuration information in XPDDDATA.DB. See Chapter 3 for information on XPDDDATA.DB.

The Configuration Check List

This section outlines the process of configuring a database for use with a client application. The steps are explained in more detail later in this chapter. This section assumes that you already know how to define a database and its objects, as described in earlier chapters of this manual. You may perform configuration tasks either at the same time you are defining or after you have already defined a database and its objects.

- ☐ Install the database and its dimensions and measures. See the section “Installing Databases, Dimensions, and Measures” later in this chapter.
- ☐ If you have a dimension or a measure that you want to hide from client application users, you can use Administrator to do this. See the section “Hiding Dimensions and Measures” later in this chapter.
- ☐ If you have installed a database, a dimension, or a measure and find that your client application users no longer have use for it, you can uninstall it. See the section “Uninstalling Databases, Dimensions, and Measures” later in this chapter.
- ☐ If you want client application users to have convenient ways of selecting and displaying data, you can use Administrator to specify hierarchies for the data. See the section “Editing Dimension Hierarchies” later in this chapter.
- ☐ When client application users display the Selector to choose the data for dimension list boxes, tables, and graphs, they often find that they are choosing the same group of dimension values again and again. You can use Administrator to save such a group of dimension values to make it easier for users to select them. See the section “Defining a Dimension’s Selections” later in this chapter.
- ☐ If a given pair of dimensions has more than one relation or conjoint, you can use Administrator to specify which one the Selector in the client application will use as the default. See the section “Specifying Default Relations” later in this chapter.
- ☐ If you would like to develop a program that the client application will run automatically every time it closes, you can use Administrator to do

this. See the section “Defining System Shutdown Programs” later in this chapter.

Installing Databases, Dimensions, and Measures

You can customize the way databases, dimensions, and measures appear to your client application users by using Administrator to install them. When you install them, Administrator creates other objects to support the customization. These objects are the Data Dictionary structures.

To see the Data Dictionary structures that Administrator creates in a database, look for objects whose names begin with the *Xn* prefix. The *X* stands for the first letter of the database, dimension, or measure. The *n* is a number that differentiates the objects for different databases, dimensions, or measures that start with the same letter.

For example, suppose you create a dimension called **TERRITORY**. When you select **TERRITORY** in the Database Browser, its properties and their values are displayed in the Inspector. When you install **TERRITORY**, Administrator generates an ID Code (with a value, for example, of T8), a LongLabelVar (T8.LONGLABEL), and a ShortLabelVar (T8.SHORTLABEL), all of which are displayed in the Inspector.

All Data Dictionary objects are automatically assigned a name that begins with the ID code, in this example, “T8.” Other objects may be generated in addition to the ones displayed in the Inspector. To view these other Data Dictionary object names in the Database Browser, choose **All Objects** from the View menu. You can use these objects in your client application code to display short and long labels for dimensions and measures in the client application you write.

If the client application is a prebuilt client application from Oracle, such as the Excel Add-In, just install the dimensions or measures. The client application does the rest.

You generally install databases, dimensions, and measures as part of the process of structuring data. The installation of these objects is described in detail in Chapter 5 of this manual.

Hiding Dimensions and Measures

You can use the Data Dictionary structures that Administrator automatically creates to display your database's dimensions and measures in a client application you write.

If you want to hide dimensions or measures from users, you can use Administrator to do this. Later, if you change your mind, you can use Administrator to show them again.

Note that if your users use the Excel Add-In as the client, you can only hide measures. By default, the Measures list in the Excel Add-In's Express Wizard displays the user-visible measures in your database. You can prevent any measure from being displayed in the Measures list.



To hide a dimension or measure:

1. Edit the dimension or measure.
2. In the editing dialog box, clear the **User Visible** box.



For more information on hiding measures, search for the following topics in the Administrator Help system:

- “Editing Formulas”
- “Editing Relations”
- “Editing Variables”

Uninstalling Databases, Dimensions, and Measures

If you find that your client application users no longer need certain databases, dimensions, or measures, you can uninstall them. However, uninstalling such objects has serious implications. See the following sections for information.

Uninstalling a Database

If you previously installed a database and you no longer need it, you might want to uninstall it.

When you uninstall a database, Administrator deletes all of the database's properties and all of the data structures it added to the database for installed dimensions and measures. You can remove an installed database and its contents from the user's view without uninstalling it by using Administrator to set the database's User Data property to NO.

When you uninstall a database, Administrator deletes the language dimension if there are no objects dimensioned by it. If you have used the language dimension to provide multilanguage support, Administrator does not delete the dimension because there are objects dimensioned by it.

Note: You should not uninstall a database unless you are certain that you will never want to use it with a client application again.

You can uninstall a database only if you have opened it in Administrator. You cannot delete databases in Administrator. You should always uninstall a database in Administrator before deleting the database file.



For information on uninstalling databases, search for the following topic in the Administrator Help system:

“Uninstalling Databases”

Uninstalling a Dimension or Measure

If you previously installed a dimension or measure but now no longer have a use for it, you can uninstall it. This way its Data Dictionary structures will not clutter your database. You must uninstall a dimension or measure before deleting it from a database, in order to keep the Data Dictionary current.

Uninstalling does not delete the dimension or measure; however, the procedure does have serious consequences. When you uninstall a dimension or measure, Administrator deletes all the Data Dictionary structures it created in your database for the dimension or measure. For example, a dimension's parent relation, which might have taken considerable effort to populate, will be deleted. Do not uninstall a dimension or measure unless you are certain you no longer need it in the client application.

You can remove an installed dimension or measure and its contents from the user's view without uninstalling it by using Administrator to set the dimension's or measure's User Data property to NO.



For information on uninstalling dimensions and measures, search for the following topic in the Administrator Help system:

“Uninstalling Database Objects”

Editing Dimension Hierarchies

The client application can use dimension hierarchies to enable users to select and display data conveniently.

The following sections present an overview of dimension hierarchies and describe the procedures you use to set them up and delete them.

Overview of Hierarchies

A hierarchy is a way of organizing a dimension's values and of providing for drilling in tables and graphs. If the client application is the Excel Add-In, the hierarchy provides a drilling in an Excel spreadsheet. The values are grouped into *levels*, with the lower levels aggregating into the higher ones. Figure 8.1 presents a hierarchy that organizes the PRODUCT dimension in the XADEMO database.

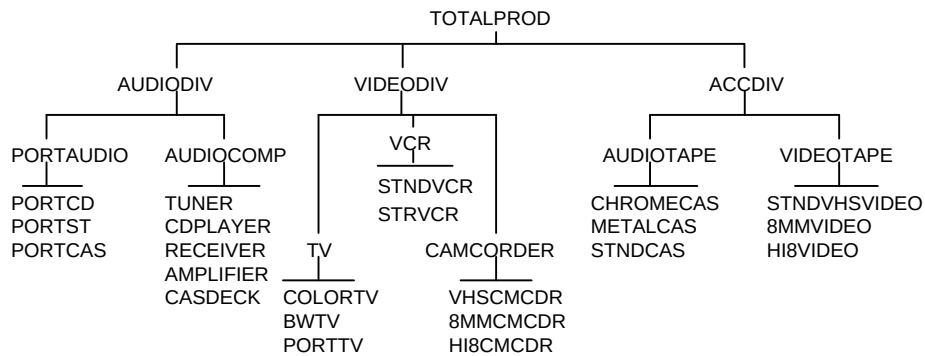


Figure 8.1 STANDARD hierarchy for the PRODUCT dimension.

The PRODUCT dimension hierarchy in Figure 8.1 has five levels. The lowest is the detail level, which is made up of dimension values such as TUNER and COLORTV. The three higher levels represent aggregations of the levels below. The TOTALPROD value, at the top of the hierarchy, is referred to as the “root.”

The PRODUCT dimension is an *embedded-total dimension*, because its values include all the values shown in Figure 8.1 — the detail values *and* the aggregate values. The aggregate values are not automatically rolled-up (or calculated), but rather are specific data values. The first 10 values of the PRODUCT dimension are listed below:

TOTALPROD
AUDIODIV
PORTAUDIO
PORTCD
PORTST
PORTCAS
AUDIOCOMP
TUNER
CDPLAYER
RECEIVER

The purpose of creating a hierarchical dimension is to allow client application users to choose groups of dimension values (by level or family) when they are selecting and displaying data. However, an embedded-total dimension alone does not give the client application enough information to allow it to know which dimension values belong to the various levels. This information is kept in a *parent relation*, which records the parent for each value in the embedded-total dimension. A value's parent is at the level immediately above that value in the hierarchy structure. For example in Figure 8.1, the diagram shows that the parent for PORTCD is PORTAUDIO, and the parent for PORTAUDIO is AUDIODIV. The value at the root of the hierarchy (TOTALPROD in Figure 8.1) has no parent.

The first 10 values of the PROD dimension are listed again below. This time, the associated values in the parent relation are listed to the right, and each product is indented to indicate its level.

Dimension Value	Parent
TOTALPROD	NA
AUDIODIV	TOTALPROD
PORTAUDIO	AUDIODIV
PORTCD	PORTAUDIO
PORTST	PORTAUDIO
PORTCAS	PORTAUDIO
AUDIOCOMP	AUDIODIV
TUNER	AUDIOCOMP
CDPLAYER	AUDIOCOMP
RECEIVER	AUDIOCOMP

An indented list is easier to understand, because all the values at a given level are positioned at the same *depth*, and children are placed deeper than their parents. The *order* of the dimension values also clarifies the relationships in the hierarchy. Furthermore, a dimension is easier to understand when *siblings* (values that share a parent) appear together, either above or below their parents.

When you set up a hierarchy for a dimension, you give the client application the information it needs to present a dimension in this way in the Selector, in dimension list boxes, and in tables. If the client application is the Excel Add-In, it presents a dimension hierarchically in the Selector and in Excel spreadsheets.

When you have configured a database a client application and you have specified dimensions as hierarchical, those dimensions are embedded-total dimensions. Administrator generates a set of Data Dictionary structures (e.g., a parent relation) that hold information about characteristics such as parent-child relationships and sibling order.

A configured database is not limited to one hierarchy for a given dimension. For example, Figure 8.1 presents a hierarchy that can be considered the standard one for the PRODUCT dimension in the XADEMO database. Figure 8.2 presents a second hierarchy that divides the same detail-level products into product types.

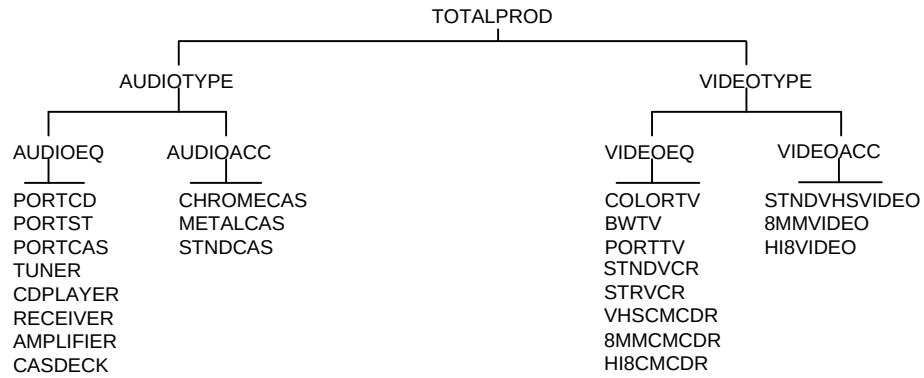


Figure 8.2 PRODTYPE hierarchy for the PRODUCT dimension.

To support this second hierarchy, you would add AUDIOTYPE, AUDIOEQ, AUDIOACC, VIDEOTYPE, VIDEOEQ, and VIDEOACC as values in the PRODUCT dimension. In addition, the database would need modifications to the Data Dictionary structures in order to accommodate the second hierarchy. For example, the parent relation for the second hierarchy would have to be filled in.

A dimension value is part of a hierarchy only if it meets one of the following criteria:

- It has a parent (except for the root)
- It has at least one child
- It has an ordering number

The following section describes the steps required for setting up hierarchies in a database.

Setting Up Hierarchies

If the client application is to display dimension values organized by hierarchies, you must use Administrator to set up the Data Dictionary structures to support those hierarchies.

You perform the following steps when you set up the hierarchy (or hierarchies) for a dimension:

1. Prepare the dimension, giving it embedded-total values.
2. Install one or more hierarchies.
3. Specify the parent-child relationships, which define hierarchies, and calculate the hierarchy.
4. Edit the hierarchy characteristics (optional).

The sections that follow describe these steps.

Remember that when you define variables dimensioned by hierarchical dimensions, you must roll up data to all levels of the hierarchy. See Chapter 4 for an overview of using hierarchical dimensions with variables.

Step 1: Preparing the Dimension

If your dimension is simply a list of detail-level items, such as products, stores, or cities, add the dimension values that provide the parents' aggregates for these values. When you have completed this step, you have turned your dimension into an embedded-total dimension. See the previous section "Overview of Hierarchies" for an introduction to embedded-total dimensions.

For example, if the PRODUCT dimension contained only the values at the lowest level of the hierarchy in Figure 8.1, you would add the values in the following list. These are aggregate values that you can calculate from the detail-level values.

TOTALPROD	VCR	AUDIODIV
CAMCORDER	PORTAUDIO	ACCDIV
AUDIOCOMP	AUDIOTAPE	VIDEODIV
VIDEOTAPE	TV	

**To add the aggregate values manually:**

If you want to type in the values by hand, display the Edit *Dimension* dialog box and choose **Add** to add the necessary dimension values. Search the Help system for the topic “Editing Dimensions.”

or

If you have a file containing these values, then you can import them from the file. See Chapter 6 for information on importing data.

For any additional hierarchies, add all the aggregate values that you need.

When you provide the data for these values, you might use the Express ROLLUP command to aggregate the detail-level data automatically. If you use the ROLLUP command to provide the aggregate data, you must first create the parent relation, as described later in Step 3; you use the parent relation as an argument to ROLLUP.

To use an Express command, first choose **Express Command** from the Tools menu to display the Express Command dialog box. You can then execute Express commands in the upper window of the command dialog box and see any results displayed in the lower window. For more information, see Chapter 4.

Step 2: Installing One or More Hierarchies

When you install a hierarchy, Administrator creates the Data Dictionary structures required to support it. By default, Administrator creates one hierarchy, called “STANDARD.” To see the Data Dictionary structures for hierarchies that Administrator creates in a database, look for objects whose names begin with the *Xn* prefix. The *X* stands for the first letter of the dimension. The *n* is a number that differentiates the objects for different dimensions that start with the same letter.

**To install a hierarchy:**

1. Define the dimension if you have not already done so.
2. Double-click on the dimension in the Database Browser to display the Edit *Dimension* dialog box.

3. On the General tab, verify that the **Install** and **Hierarchical** boxes are selected; select them if they are not.
4. On the Values tab, choose **Add** in the Hierarchy Info box to display the New Hierarchy dialog box. Type a name for the new hierarchy and choose **OK**.
Tip: If you want to use the name Standard for a hierarchy, you can omit Step 4 and edit the existing Standard hierarchy.
5. In the Description box, type in a descriptive name for the hierarchy. Client application users will see this descriptive name. If you do not specify a descriptive name, client application users will see the hierarchy name instead.
6. If the client application is an Oracle prebuilt client application, such as the Excel Add-In, and you want this hierarchy to be the default in the Selector, select **Default**.
7. To add more hierarchies, repeat Steps 4 through 6.
8. Choose **Close** when you have finished adding hierarchies.

Step 3: Specifying the Parent-Child Relationships and Calculating Hierarchy Data

The earlier section “Overview of Hierarchies” described the parent-child relationship in a hierarchy. It also explained that the parent relation, *Xn.PARENT*, is the Data Dictionary structure that holds parent-child information. You supply *Xn.PARENT* values to indicate the parent of every value in your dimension.



To specify the values of the parent relation:

1. Double-click on the dimension in the Database Browser to display the Edit *Dimension* dialog box.
2. Select the Values tab.
3. In the Hierarchy box, select the hierarchy for which you want to specify the values of the parent relation.
4. Specify the parent by doing one of the following:

At the top of the Values tab, for each dimension value, supply the name of its immediate parent. You can use the drop-down list to select the appropriate value. If a value has no parent, either because it is at the top of the hierarchy or because it is not in the hierarchy, leave the parent blank.

5. Choose **Calculate** to display the Calculate Hierarchy Structures dialog box, in which you specify whether to calculate for the current hierarchy or for all hierarchies.
6. Select **Calculate for the current hierarchy** or **Calculate for all hierarchies** and choose **OK**.
7. Choose **Close** when you have finished calculating hierarchy data.

Tip: Instead of using steps 1 through 4, you can use a data loader program to import the parent values into the parent relation. See Chapter 6 for information on importing data.

Step 4: Editing the Hierarchy Characteristics

The earlier section “Overview of Hierarchies” explains that the levels of a hierarchy and the order of values in a dimension make it easy for client application users to select and view data. If you want, you can provide descriptions for the client application to use for the levels, and you can specify the order of the dimension values.

If the client application is an Oracle prebuilt application, such as the Excel Add-In, these level names appear in the Selector’s Level and Family tools.



To specify level descriptions and order:

1. Double-click on the dimension in the Database Browser to display the Edit *Dimension* dialog box.
2. Select the Values tab.
3. In the Hierarchy box, select the hierarchy for which you want to specify the levels.
4. Choose **Levels** to display the Hierarchy Levels dialog box. Check that the number of levels shown in the Levels column meets your

expectations; if not, return to the Edit *Dimension* dialog box, correct the information in the Parent column, and recalculate the hierarchies.

5. To enter a level description, double-click on a cell in the Description column of the Hierarchy Levels dialog box. You will enter edit mode. Type in the description.

The level descriptions are stored in the *Xn.LEVELDESC* variable. Because *Xn.LEVELDESC* is dimensioned by the language dimension, each level can have a different description for each language in your system.

6. After providing descriptions for all levels, choose **OK** to return to the Edit *Dimension* dialog box.
7. If you want to sort the dimension values in a certain way as you work with them in the Edit *Dimension* dialog box, choose **Sort** to display the Sort Dimension Values dialog box. Make the following settings:
 - a. In the Field box, select whether to sort by database order, dimension value, parent, short name, or long name.
 - b. In the Order box, select whether to sort in ascending or descending order.
 - c. Choose **OK** after making the specifications.
8. In the Drill Direction box, select **Down** if you want the client application to show children below their parents, or select **Up** if you want the client application to show children above their parents.
9. To specify the order of dimension values, in the Order column at the top of the tab, type in integers or decimals.
10. Recalculate the hierarchy data as described in the earlier section “Step 3: Specifying the Parent-Child Relationships and Calculating Hierarchy Data.” You must calculate the data at this point to register the changes you have made.
11. Choose **Close** when you have finished.

Deleting a Hierarchy

Think carefully before you delete a hierarchy. When you delete it, all the data structures, such as the parent relation and description variables, are modified to delete any information that applied to the hierarchy. Delete a hierarchy only if you are sure you have no further use for it.



To delete a hierarchy:

1. Double-click on the dimension in the Database Browser to display the Edit *Dimension* dialog box.
2. Select the Values tab.
3. In the Hierarchy box, select the hierarchy you want to delete.
4. In the Hierarchy Info box, choose **Delete**. You are prompted to verify that you want to delete the hierarchy. Choose **Yes**.
5. Choose **Close** when you have finished.

Defining a Dimension's Selections

When client application users display the Selector to choose the data to display in the client application, they often find that they are choosing the same group of dimension values again and again. As part of the configuration of a database, you can save such a group of dimension values as a *saved selection*. The Selector recognizes these saved selections and lets users choose them. Client application users can also save their own selections.

The selections you save in Administrator will be available in the client application's Selector.

The two types of saved selections are *value* and *script*. In the client application, the two types are called *static* and *dynamic*. The setting of the selection depends on its type:

- The setting of a value selection is a multiline list of dimension values. A value selection is static; if you delete one of the values that is listed,

or if you add a dimension value that should be included, you must change the list yourself.

- The setting of a script selection is either the name of an Express program or a series of program lines. A script selection can be dynamic, depending on how you write the code for the script. For example, if your program uses a command to limit the PRODUCT dimension to the top 10 values based on SALES, the actual values in the selection will differ based on the sales values.

If the dimension has one or more hierarchies associated with it, you might want the selected values limited to only the values that belong to the hierarchy, and you might want the values sorted according to the order associated with the hierarchy.

To define saved selections, you use the Selections dialog box, which you display by choosing **Selections** from the Dictionary menu. If you have defined a saved selection that client application users no longer need, you can use the Selections dialog box to delete it.



For more information on saved selections, search for the following topics in the Administrator Help system:

- “Defining Saved Selections”
- “Deleting Saved Selections”
- “Selections Dialog Box”

Specifying Default Relations

If your users are using an Express client application, they will have access to the Selector. The Selector provides several tools for selecting data. One of these is the Attribute tool, which lets the user choose dimension values based on an Express relation. This section provides an example of a relation and describes how to specify a default relation.

The following example from the XADEMO database is the QUARTER_TIME relation, which relates the TIME and QUARTER dimensions.

```
report w 12 quarter_time
```

TIME	QUARTER_TIME
-----	-----
JAN95	Q1.95
FEB95	Q1.95
MAR95	Q1.95
APR95	Q2.95
MAY95	Q2.95
JUN95	Q2.95
JUL95	Q3.95
AUG95	Q3.95
SEP95	Q3.95
OCT95	Q4.95
NOV95	Q4.95
DEC95	Q4.95
JAN96	Q1.96
FEB96	Q1.96
MAR96	Q1.96
APR96	Q2.96
MAY96	Q2.96
.	.
.	.
.	.

A relation, such as QUARTER_TIME, provides a way to choose the values of one dimension based on the values of another dimension. For example, by referencing QUARTER_TIME you can select all the months in a given quarter, or you can select the quarter that contains a given month. The relationship goes both ways.

If a given pair of dimensions has only one relation or conjoint, the Selector provides the Attribute tool based on that relationship. For each of the two dimensions, you can use the Attribute tool to choose values based on the other dimension. You do not have to take any configuration steps to make this happen, except that both dimensions should already be installed so that their useful labels appear. You can also use the IsAttribute property to specify whether a relation or conjoint dimension is considered an attribute by the Selector.

If a given pair of dimensions has more than one relation or conjoint, you should use Administrator to specify which one the Selector will use as its

default. If you do not make this specification, the Selector chooses a default relation or conjoint based on internal, runtime conditions.

To specify default relations, you use the Default Relation dialog box, which you display by choosing **Default Relation** from the Dictionary menu. Table 8.1 describes the settings you specify in this dialog box.

Table 8.1 Default Relation Information

Settings	Description
Relation ID	An identifier assigned by Administrator.
Dimension 1	One related dimension.
Dimension 2	The other related dimension.
Relation	A list of the relations or conjoint dimensions that relate Dimension 1 to Dimension 2. You select the one that will be the default.
Database	The database in which the related dimensions reside. Administrator automatically supplies this value when you specify one of the two dimensions.

You ordinarily specify only one default relation for a given pair of dimensions. However, in a different situation, you could specify the same pair of dimensions twice in the Default Relation dialog box, but in different order. For example, if you had a STAFF dimension and a DISTRICT dimension, you could make the following specifications:

Dimension 1	Dimension 2	Relation
STAFF	DISTRICT	DISTRICT_STAFF
DISTRICT	STAFF	STAFFINFO

In this example, the Selector will use the DISTRICT_STAFF relation to provide the Attribute tool for STAFF, but it will use the STAFFINFO conjoint to provide the Attribute tool for DISTRICT. Keep the following guidelines in mind:

- If you specify a single relation for a given pair of dimensions, the Selector uses it to provide attributes for both dimensions.
- If you specify two relations for a given pair of dimensions and their order in the box is reversed, the Selector uses the specified relation to provide the attributes for the dimension you specified in the Dimension 1 box.

- Do not specify two different relations for a given pair of dimensions using the same order. This would provide conflicting specifications; the Selector would use the first of the two relations to provide attributes for both dimensions.



See Chapter 1 for information on relations. See Appendix B for information on conjoint dimensions.



For more information on default relations, search for the following topics in the Administrator Help system:

“Identifying Default Relations”

“IsAttribute Property”

Defining System Shutdown Programs

You can develop a program that the client application will run automatically every time it closes. You might want to run a program that disconnects the user from a remote system, reverses the operations of a system preprocessor, or does other similar activities.

The system has only one shutdown program. You must ensure that the database containing the shutdown program is attached so that the client application can find the program when it is ready to shut down. You can name the program anything you like. You use the System Shutdown Program dialog box to specify which program to use as the shutdown program.



For more information on system shutdown programs, search for the following topics in the Administrator Help system:

“Defining Programs”

“Editing Programs and Models”

“Specifying a System Shutdown Program”

A Simple Case Study

This section describes the basic steps involved in importing data from a flat file and putting it into a form that is easy to use in the client application. These basic steps are to:

1. Understand the data.
2. Create an Express database and its objects.
3. Import the data.
4. Configure the database.

This case study is meant to give you a basic overview of the process of creating an Express database. See Chapter 4 for information about the actual procedures for creating a database and its objects. See Chapter 6 for information about importing data.

You can also import data or obtain data dynamically from relational databases. Chapter 5 describes how to use dynamic SQL reach-through.

Step 1: Understand the Data

First, you need to understand the format of the data in the flat file. Is the data structured in the file, or does it appear in raw columns? What does each column or field represent? Figure 8.3 shows an ASCII file of sales data. It has five fields, which show month, product, geography, units, and dollar sales.

"JAN90	", "TENTS	", "TOTUS	"	1429,	20156
"JAN90	", "TENTS	", "EAST	"	453,	5183
"JAN90	", "TENTS	", "BOSTON	"	200,	1938
"JAN90	", "TENTS	", "ATLANTA	"	253,	3245
"JAN90	", "TENTS	", "CENTRAL	"	478,	9316
"JAN90	", "TENTS	", "CHICAGO	"	181,	3418
"JAN90	", "TENTS	", "DALLAS	"	297,	5898
"JAN90	", "TENTS	", "WEST	"	498,	5657
"JAN90	", "TENTS	", "DENVER	"	227,	1205
"JAN90	", "TENTS	", "SEATTLE	"	271,	4452
"JAN90	", "CANOES	", "TOTUS	"	1596,	20106
"JAN90	", "CANOES	", "EAST	"	607,	5213

Figure 8.3 A structured ASCII file showing sales and units data.



See the *Express Language Programming Guide* for information about structured and ruled ASCII files.

Step 2: Create an Express Database and Its Objects

Once you understand the structure of the file and of the data, you create an Express database and structure it to hold the data. The structured ASCII file in Figure 8.3 requires a Time dimension, a Product dimension, and a Geography dimension. It also requires two variables, one for Units and another for Sales. Each variable is dimensioned by the Time, Product, and Geography dimensions. The Geography data appears to be hierarchical, so you should define the Geography dimension with a hierarchy.

Administrator automatically creates a dimension for language support when you install a database; this dimension is named *XXX.LANGDIM*, where *XXX* stands for the database name. Also, when you install a dimension, Administrator automatically creates supporting objects for storing labels. When you create a hierarchical dimension, Administrator creates additional variables and relations for supporting the hierarchy. When you choose **All Objects** from the View menu, these supporting objects appear in the Database Browser.



See Chapter 4 for more information about creating Express database objects.



For information about creating Express database objects, search for the following topics in the Administrator Help system:

- “Creating New Express Databases”
- “Defining Dimensions”
- “Defining Formulas”
- “Defining Hierarchies”
- “Defining Models”
- “Defining Programs”
- “Defining Relations”
- “Defining Valuesets”
- “Defining Variables”

Step 3: Import the Data

After you create all the dimensions and variables you need, you import the data. Administrator provides dialog boxes to guide you through this process. Figure 8.4 shows the Import ASCII Data dialog box, filled in with the necessary information to import the data shown in Figure 8.3.

Field #	Sample Text	Express Object
Field 1	JAN90	APPEND month
Field 2	TENTS	APPEND product
Field 3	TOTUS	APPEND geography
Field 4	1429	units
Field 5	20156	sales

File Sample:
 "JAN90" ,"TENTS" ,"TOTUS"

Processing Info
 Starting record: 1 Ending record: EOF Update every n records: 5000

Figure 8.4 The Import ASCII Data dialog box, with dimensions and variables specified for each field in the ASCII file.



See Chapter 6 for more information about importing data.



To see instructions for importing data, search for the following topics in the Administrator Help system:

“Importing ASCII Data”

“Importing Data from Relational Databases”

“Importing EIF Files”

Step 4: Configure the Database

Once the data is in the Express database, you provide any other structures or information necessary for making the database easy to use. For example, you can create hierarchies for a dimension. Figure 8.5 shows the Edit *Dimension* dialog box with hierarchical relationships specified for the Geography dimension.

The screenshot shows the 'Edit GEOGRAPHY' dialog box. At the top, the 'Name' field is set to 'geography'. Below this are three tabs: 'General', 'Labels', and 'Values', with 'Values' currently selected. The main area contains a table with 'Child' and 'Parent' columns, showing a hierarchy where 'TOTUS' is the parent of 'EAST', 'BOSTON', 'ATLANTA', 'CENTRAL', 'CHICAGO', 'DALLAS', and 'WEST'. 'EAST' is the parent of 'ATLANTA' and 'CENTRAL', and 'CENTRAL' is the parent of 'CHICAGO' and 'DALLAS'. To the right of the table are 'Add...', 'Delete...', and 'Sort...' buttons. Below the table is a 'Hierarchy Info' section with a 'Hierarchy' dropdown set to 'STANDARD', a 'Description' field, and 'Add...' and 'Delete...' buttons. There is also a 'Drill Direction' dropdown set to 'Down'. At the bottom right, the 'Database' dropdown is set to 'SIMPLE'. A 'Close' button is located at the top right of the dialog.

Child	Parent
TOTUS	
EAST	TOTUS
BOSTON	EAST
ATLANTA	EAST
CENTRAL	TOTUS
CHICAGO	CENTRAL
DALLAS	CENTRAL
WEST	TOTUS

Figure 8.5 Hierarchical relationships for the Geography dimension.

In addition to setting up hierarchies, you can specify meaningful labels for the Express objects. These labels can help clarify information in the Selector, in dimension list boxes, and in tables and graphs. Figure 8.6 shows sample labels for the Geography dimension.

The screenshot shows a dialog box titled "Edit GEOGRAPHY". At the top, there is a "Name:" field containing "geography" and a "Close" button. Below this are three tabs: "General", "Labels", and "Values". The "Labels" tab is currently selected. Inside the "Labels" tab, there are four text input fields arranged in two rows. The first row contains "Long Desc:" with the text "Geographical areas in which we sell products" and "Short Label:" with "Geog. Area". The second row contains "Short Plural Label:" with "Geog. Areas" and "Long Label:" with "Geographical Area". Below these, there are two more fields: "Long Plural Label:" with "Geographical Areas". At the bottom right of the dialog, there is a "Database:" label and a dropdown menu showing "SIMPLE".

Figure 8.6 Edit *Dimension* dialog box, with labels for the Geography dimension.

You should specify a description and long and short names for the values of each dimension. For example, you might give the TOTUS dimension value the long name “Total for United States” and the short name “Total US”.

You can also configure the database by doing the following:

- Installing objects that you want client application users to see
- Hiding measures from client application users
- Defining saved selections that contain the dimension values client application users want to display in a table or graph
- Specifying default relations



To see instructions for performing different configuration tasks, search for the following topics in the Help system:

- “Defining Formulas”
- “Defining Hierarchies”
- “Defining Models”
- “Defining Relations”
- “Defining Saved Selections”
- “Defining Valuesets”
- “Editing Dimensions”
- “Editing Formulas”
- “Editing Variables”

Viewing the Data in the Client Application

After you configure the database, exit Administrator. Start up your client application and view the data there.

For example, suppose you are using the default client application that comes with Express Server 6.0, the Excel Add-In. Start up Excel and run the Express Wizard. Look at the data to see if it appears as you expect.

Chapter 9

Managing a Database

Once you have created a new database and configured it, you can take steps to manage your database. This chapter describes:

- Using Express commands to perform a variety of database management tasks
- Improving the efficiency of your database
- Making calculations efficiently
- Planning for a smaller database
- Keeping your database efficient
- Saving a database in two or more files
- Controlling access to your database
- Summary

Using Express Commands

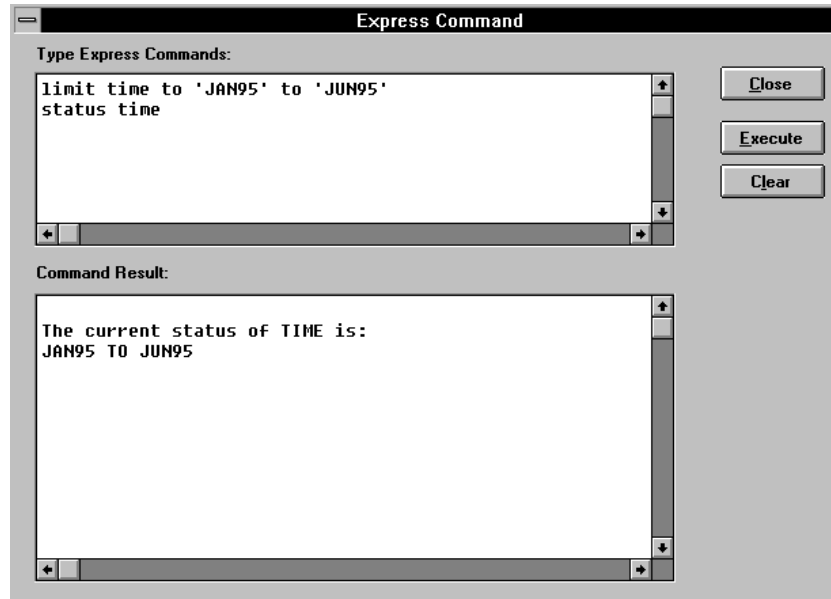
You can use certain Express commands to perform a variety of database management tasks. This section describes a few of those commands and suggests when you are likely to use them. The following procedure explains how to execute commands in Administrator.



To execute an Express command in Administrator:

1. From the Tools menu, choose **Express Command**. The Express Command dialog box is displayed.
2. Enter one of more Express commands in the upper window.

3. To execute the command you have entered in the upper window, choose **Execute**. If the command produces any output, that output is displayed in the lower window.



Using LIMIT to Change Dimension Status

You can view part of your data in the Express Command window by changing the status of dimensions. The *status* is a list of existing values you choose by using the LIMIT command.

For example, suppose your database has a SALES variable. This variable contains sales data stored by month for the past five years, but you only want to look at the sales for the past six months. You can limit the status of the MONTH dimension to include only the last 6 months, rather than all 60 dimension values. You change dimension status with the LIMIT command. If you then execute a REPORT command, the sales data for the last six months *only* is displayed in the Express Command window.

```
limit month to last 6  
report sales
```

You can also limit the dimension to a single value or a range of values by specifying the actual dimension values.

```
limit month to 'JAN95' to 'JUN95'
```

If you want to view a specific portion of an object's data in Administrator, be sure to set the status of its dimensions to 40 or fewer values. Otherwise, the dimensions are automatically limited to 40 values. For example, suppose you display the Edit a Variable dialog box. When you select the Data tab, a maximum of 40 cells in that variable are available to edit. You can use the LIMIT command to limit that variable's dimensions in order to specify the 40 (or fewer) values you want to edit.

In addition to editing variables or formulas, you may want to use LIMIT to set dimension status when you define relations, saved selections, and valuesets. If you want to export only certain values of dimensions, use the LIMIT command to set the status of those dimensions before you export them.

Using STATUS to Check the Current Status

When you use the STATUS command with a dimension name, it displays the current status for that dimension.

Dimension status does not have to be made up of consecutive values. You can specify more than one range in a LIMIT command or you can add or remove values from the current status list. The following commands show the current status of MONTH (using the STATUS command), and then add six months to the current status with the ADD keyword in the LIMIT command.

Suppose the status of the MONTH dimension is currently limited to the first six months of 1995. If you execute the following command, it displays the current status of that dimension.

```
status month
```

```
The current status of MONTH is:  
JAN95 TO JUN95
```

The following commands add values to the current status and use the STATUS command to verify the results.

```
limit month add 'JAN97' to 'JUN97'  
status month
```

The current status of MONTH is:
JAN95 TO JUN95, JAN97 TO JUN97

You can also reduce the current status with the KEEP keyword. KEEP, REMOVE, and ADD operate differently from TO:

- TO operates on all the values of the dimension, regardless of its status. If you have five years of sales data from 1993 to 1997, the command LIMIT MONTH TO FIRST 6 gives the first six months of 1993, regardless of the current status of MONTH.
- KEEP, REMOVE, and ADD operate on the current status. If the current status is all the months of 1997, the command LIMIT MONTH KEEP FIRST 6 gives the first six months of 1997.

Suppose you first check the status of MONTH, which produces the following results.

```
status month
```

The current status of MONTH is:
JAN97 TO DEC97

If you limit the dimension and check its status, the STATUS command shows the status is now limited to the dimension's first six values.

```
limit month to first 6  
status month
```

The current status of MONTH is:
JAN93 TO JUN93

Use the following commands to limit the status to the first 6 months of 1997.

```
limit month to year 'YR97'  
limit month keep first 6  
status month
```

The current status of MONTH is:
JAN97 TO JUN97

One example of when you may need to use LIMIT in this way is when you map SQL data to an Express database. You will use a selection preprocessor script (as described in detail in the Administrator Help system). In that script, you can include any valid Express command or program name, but it is likely you will use LIMIT commands with the ADD keyword.



Refer to the LIMIT command in the Express Language Help system for more details about using this command.

Changing Dimension Status to All

When you wish to set the status of a dimension back to all of its values, you can use the LIMIT command with the ALL keyword. For example, you can use the following command to set the status of the MONTH dimension back to all of its values.

```
limit month to all
```

When you want to set the status of all dimensions to all of their values, you can use the ALLSTAT command. To do so, type `allstat` in the upper window of the Express Command dialog box, then choose **Execute**.

Improving Database Efficiency

When you develop a complex database, you need to know how the choices you make affect the efficiency of the database. This section describes methods for:

- Storing data in the least amount of disk space
- Performing computations as efficiently as possible

To plan an efficient Express Server database, you need to consider how you will use the database and how Express Server interacts with the rest of the

computer system. For example, the way you define dimensions and variables affects the size of the database file, the efficiency of calculations, and the amount of memory the database requires.

The following list of questions illustrates the points you need to consider and indicates which section of this chapter you should read:

1. How are your users likely to display your data?

The section entitled “Making Calculations Efficiently” describes how to make the list of dimensions in a variable’s definition correspond to the way you think your users are most likely to display your data.

2. Do you make calculations whose results are not needed when your Express Server session is over?

The section entitled “Planning for a Smaller Database” describes how to use temporary variables to save disk space.

3. Do you have variables with many empty data cells?

The section entitled “Handling Sparse Data” describes methods to keep from saving empty cells on disk.

4. Which is the least available resource on the system where the database resides: disk space, CPU time, or memory?

If disk space is scarce, you can use the techniques in the sections entitled “Planning for a Smaller Database” and “Saving a Database in Two or More Files.”

If you need to limit CPU time, keep calculated data in variables *instead* of using formulas.

If you need to conserve memory, see the section entitled “Conserving Memory While Running Express Server” for suggestions on reducing memory usage.

Making Calculations Efficiently

If the biggest constraint on your system is the time it takes to process a job, rather than how much disk space you use, you can reduce the amount of CPU time you use with the following techniques:

- Define variables with dimensions ordered to match the order in which users are likely to view data
- Save frequently needed calculations in variables

This section describes in detail how data is stored.

Data in a multidimensional variable is sorted in the order of the dimensions. Data for each value of the first dimension is stored in order in a *page*, or block of data, before data is stored. When Express Server retrieves data from a variable, it reads a page of data from disk. If you define a variable with its dimensions in the order a user is likely to display the data, Express Server can retrieve the data most efficiently by getting many values from the same page, thereby reading fewer pages.

Figure 9.1 gives an example of how data is stored. UNITS is a variable with an integer data type that is dimensioned by MONTH, PRODUCT, and DISTRICT. The data is described with a code in which D means DISTRICT, which is the slowest-varying dimension; P means PRODUCT; and M means MONTH, which is the fastest-varying dimension. In the list, the code D1P1M1 means the UNITS value for the first district, first product, and first month. All the months for the first product and district are stored in order; followed by the months for the second product of the first district, and so on.

D1P1M1	D1P1M2	D1P1M3	D1P1M4	D1P1M5	D1P1M6	D1P1M7	D1P1M8	D1P1M9	...	D1P1Mnn
D1P2M1	D1P2M2	D1P2M3	D1P2M4	D1P2M5	D1P2M6	D1P2M7	D1P2M8	D1P2M9	...	D1P2Mnn
D1P3M1	D1P3M2	D1P3M3	D1P3M4	D1P3M5	D1P3M6	D1P3M7	D1P3M8	D1P3M9	...	D1P3Mnn
D2P1M1	D2P1M2	D2P1M3	D2P1M4	D2P1M5	D2P1M6	D2P1M7	D2P1M8	D2P1M9	...	D2P1Mnn
D2P2M1	D2P2M2	D2P2M3	D2P2M4	D2P2M5	D2P2M6	D2P2M7	D2P2M8	D2P2M9	...	D2P2Mnn

Figure 9.1 How dimensioned data is stored.

If you define a variable so that your users use the default layout with the variable's first dimension across the columns of tables or spreadsheets in client applications, Express Server can retrieve groups of consecutive values from each page of data. For UNITS in Figure 9.1, a client application can retrieve a group of month values before calculating where to find the next data. This does not mean users should never use a different layout; it just takes more processing time.

If you have data variables or formulas that share the same dimensionality, you should define these variables or formulas in the same dimension order. The data is retrieved the same way for each variable, thus increasing processing speed. This is a good first step toward optimization.

Saving Calculated Data

The second recommendation for efficient use of processing time is to save frequently used calculations in variables. This, of course, increases the size of your database. However, if you save a calculation, all you have to do is retrieve it, rather than recalculate it.

You can perform this task using Express commands. For example, suppose the variable SALES is simply the product of UNITS multiplied by PRICE. You could define a formula that is calculated every time you refer to SALES.

```
DEFINE SALES FORMULA UNITS * PRICE
```

However, you could alternatively define a variable and set it equal to UNITS times PRICE.

```
DEFINE SALES VARIABLE DECIMAL <MONTH PRODUCT DISTRICT>  
SALES = UNITS * PRICE
```

This variable does add to the size of your database but it avoids the need to recalculate SALES again unless, of course, you have to correct some of your data. SALES, as a formula, is self-correcting.

Planning for a Smaller Database

When you first define a variable or relation, its values are all NA by default. *NA* means that the cell has either never had a value assigned to it or that a calculation produced an NA result. A *cell* is a single data value of an expression. In a dimensioned expression, a cell is identified by one value from each of the dimensions of the expression. For example, if you have a variable with the dimensions MONTH, PRODUCT, and DISTRICT, each

combination of a month, a product, and a district identifies a separate cell of that variable.

If you have data in each variable for every combination of dimension values, with no NA values, consider the following options for saving disk space.

- If some of your variable data is derived from other data, you can save space by defining a formula to calculate the data instead of storing it.

To define a formula, choose **Define** from the Dictionary menu, then choose **Formula** from the submenu to display the Define a Formula dialog box. Specify the formula's name and dimensions. You may also specify the formula's data type, although it is not required. In the Equation box, type the formula, for example:

`units * price`

For more details about defining a formula, search for “Defining Formulas” in the Administrator Help system.

If you do not use the calculated data very often, the savings in disk storage can outweigh the cost of CPU time to repeatedly calculate the formula.

You should use formulas whether or not you need to conserve disk space if you make frequent corrections to your data. A formula ensures that the derived data is as current as the newly corrected data.

- If you define a variable to store interim calculations, make it a temporary variable by selecting **Temporary** in the Storage box when you define a variable. Search for “Define a Variable Dialog Box” in the Administrator Help system for more information.

Temporary variables are for data that is needed only in the current client application session; the actual data does not get stored in the database. Temporary variables only have a definition in the database file; they do not contain any associated data. When you detach the database in which any temporary variables are defined, their values are lost.

Handling Sparse Data

If a variable has a large number of NA values, it is called *sparse data*. By default, Express stores all NA cells unless an entire page of data is made up of NA values.

Data may be sparse because certain combinations of dimension values are not applicable to your company, or because data is not available for a range of values.

You can reduce the size of your database by reducing the size of the variables in which data is sparse. Sparse data comes in two varieties:

- Controlled (or structural) sparsity
- Random sparsity

Data with Controlled Sparsity

Controlled sparsity means that a range of values of one or more dimensions has no data. This is often a result of the way you design your database.

For example, suppose you define a new variable called PROJECTED_SALES. Notice in the following figure that this definition specifies MONTH as the fastest-varying dimension, because MONTH was the first dimension selected. PRODUCT is the second-fastest varying dimension, and DISTRICT is the slowest-varying dimension.

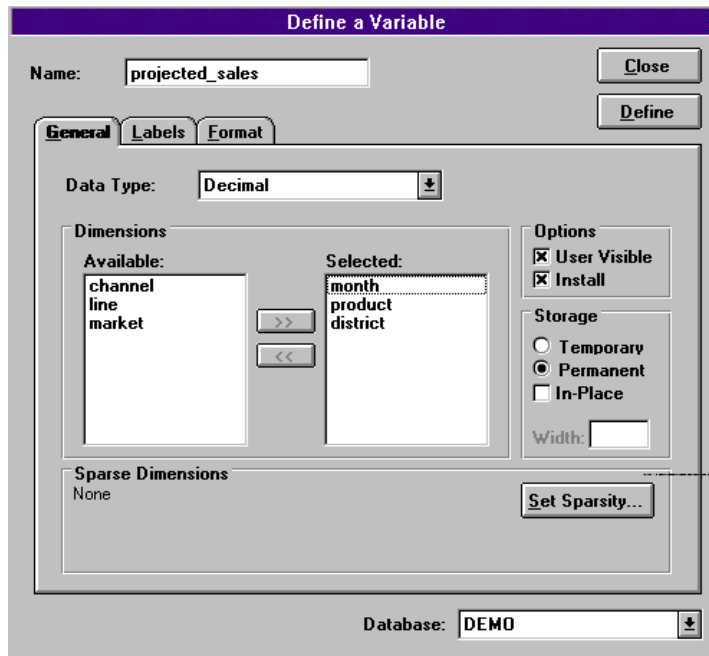


Figure 9.2 MONTH is the fastest-varying dimension.

Now suppose there are no values for the last three months. The cells exist because these months exist as MONTH dimension values, but all the cell values are NA, as shown in the following figure. All of these NA values will be saved to disk.

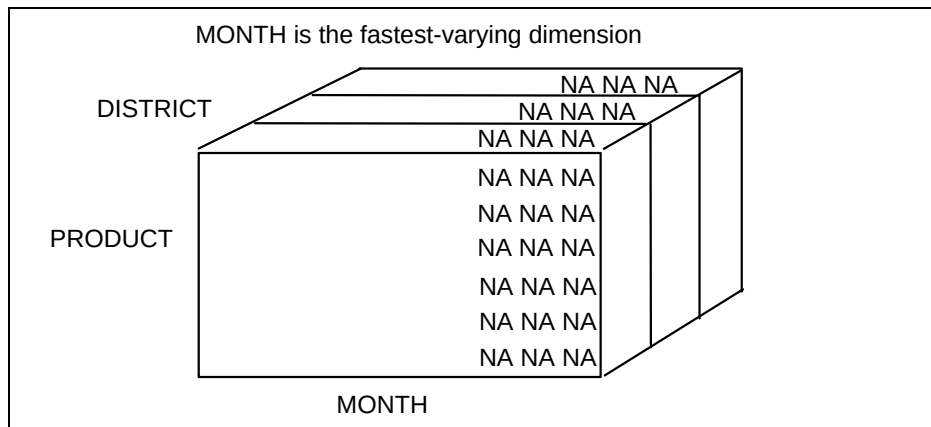


Figure 9.3 All NA data will be saved on disk.

You can save disk space for data with controlled sparsity by the way you define variables. When you define a multidimensional variable, be sure that the sparse dimension is the *last* dimension you specify. For example, when you use the Define a Variable dialog box to define the PROJECTED_SALES variable, as shown in the following figure, MONTH must be the last dimension you select and, therefore, will be the last dimension in the list of selected dimensions.

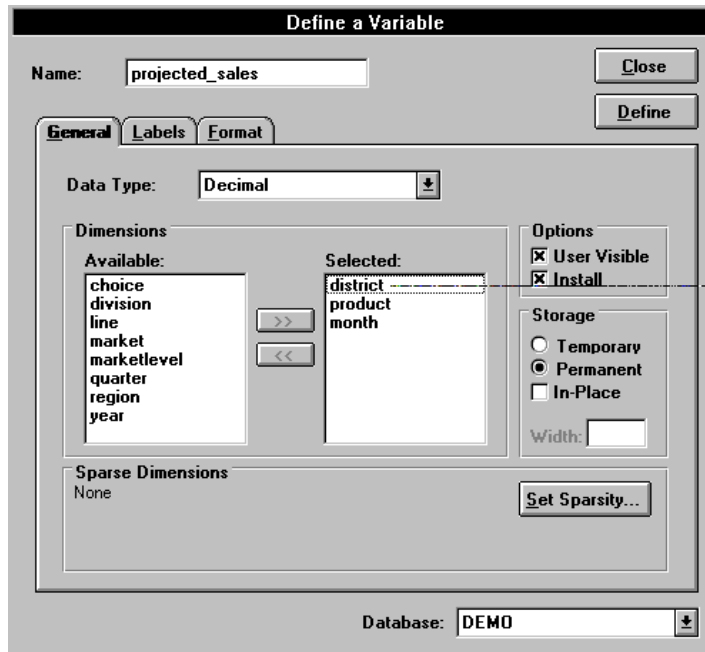


Figure 9.4 MONTH is the slowest-varying dimension.

If you specify the sparse dimension as the last dimension in the variable's definition, the NA values are grouped together on one or more data pages. Express Server does not store pages that only contain NA values, so you save space in the database.

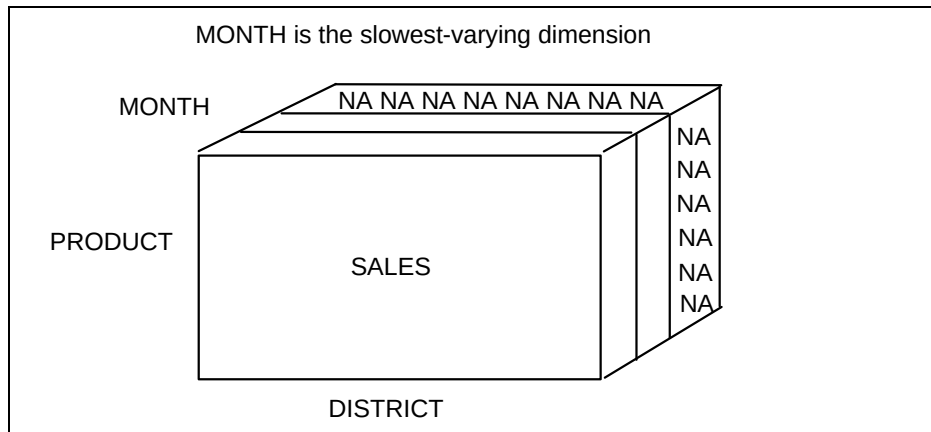


Figure 9.5 NA values are grouped on a page and will not be saved to disk.

This method saves disk space, but can cause inefficient processing if other variables have the same dimensions specified in a different order or if accessing this data causes more pages to be retrieved. The earlier section entitled “Making Calculations Efficiently” describes how this works. In using these techniques, you must decide which you would rather optimize, processing or storage.

Data with Random Sparsity

Random sparsity means that NA values are scattered throughout a variable, usually because some combinations of dimension values never have any data. This is often a result of the nature of your business.

For example, a district might only sell certain products and never have data for other products. Other districts might sell some of those products and other ones as well. The following example shows the values of a variable with sparse data and its dimensions.

This example uses a variable named SALES, which is dimensioned by MONTH, PRODUCT, and DISTRICT. The following report of SALES shows a random pattern of NA values.

-----SALES-----						
	JAN95	FEB95	MAR95	APR95	MAY95	JUN95

BOSTON:						
COFFEE	31,462.80	58,343.75	54,158.84	45,459.55	63,727.98	52,232.05
SUGAR	45,633.38	66,960.37	51,201.41	41,119.10	66,985.30	34,103.41
GRITS	NA	NA	NA	NA	NA	NA
GRANOLA	38,353.49	44,890.01	32,705.20	59,889.76	26,695.59	25,807.17
ATLANTA:						
COFFEE	62,373.42	37,698.17	39,944.58	42,854.71	31,815.47	26,659.09
SUGAR	NA	NA	NA	NA	NA	NA
GRITS	55,075.33	64,352.74	36,571.66	31,048.07	28,430.80	69,612.45
GRANOLA	NA	NA	NA	NA	NA	NA
CHICAGO:						
COFFEE	28,302.61	31,301.34	53,122.95	32,510.59	50,884.83	33,123.26
SUGAR	43,460.88	22,743.96	45,991.54	39,841.34	25,657.83	50,375.86
GRITS	NA	NA	NA	NA	NA	NA
GRANOLA	NA	NA	NA	NA	NA	NA

To eliminate the NA values and make the variable smaller, you can set sparsity when you define a variable such as SALES. When you do this, you are defining a sparse variable.

Defining Sparse Variables

To handle controlled sparsity, you simply select dimensions in the correct order when you define a variable. However, to handle random sparsity, you need to define sparse variables.

A *sparse variable* is a variable that is defined as sparse along one or more of its dimensions.



To define a sparse variable:

1. From the Dictionary menu, select **Define**, then select **Variable** from the submenu to display the Define a Variable dialog box.
2. Specify information such as the name, data type, and dimensions of the variable.
3. Choose **Set Sparsity**. The Set Sparsity dialog box is displayed, which lists the dimensions you have already selected for the variable you are defining.

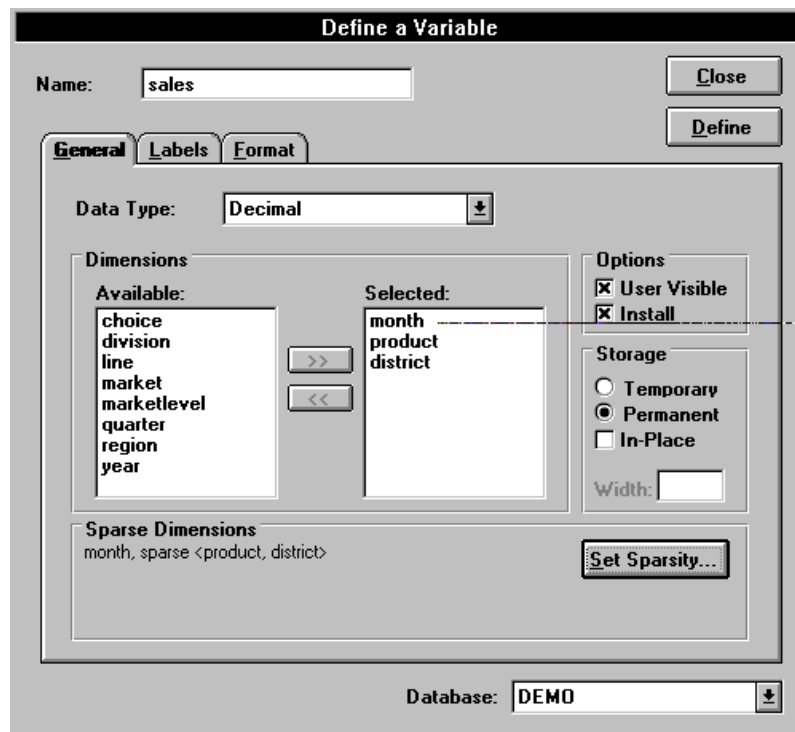
4. In the Select Dimensions box, hold down the CTRL key and select the dimensions you want to designate as sparse. When you set sparsity, the dimensions you select must be adjacent to each other in the variable's definition. In other words, they must be adjacent to each other in the Selected box in the Define a Variable dialog box. (In some cases, you may want to set sparsity for additional dimensions of the variable. See the section called "Multiple Composites," later in this chapter.)
5. Choose **Set**.
6. Choose **OK**. This closes the Set Sparsity dialog box and returns you to the Define a Variable dialog box.

Notice that the sparsity you have selected is now displayed in the Sparse Dimensions box.
7. Choose **Define** to define the variable.

Tip: The position of your sparse dimensions in the list of dimensions in the Selected box is important. If the sparse dimensions are the last dimensions in the Selected box, there are no performance implications. However, if the sparse dimensions are *not* the last dimensions in the Selected box, performance could become an issue when accessing the sparse variable. See "Using CHGDFN SEGWIDTH for Execution Efficiency" later in this chapter to learn about steps you can take to ensure good performance.

For example, suppose you are defining a sparse variable named SALES, which is dimensioned by MONTH, PRODUCT, and DISTRICT. If you select PRODUCT and DISTRICT as sparse dimensions, when you return to the Define a Variable dialog box, notice how this information is reflected in the Sparse Dimensions box near the bottom of the dialog box.

In the Sparse Dimensions box, the "month, sparse <product district>" notation means that data will be sparse along the PRODUCT and DISTRICT dimensions, but not along the MONTH dimension.



The "Define a Variable" dialog box is shown with the "General" tab selected. The "Name" field contains "sales". The "Data Type" is set to "Decimal". The "Dimensions" section shows "Available" dimensions: choice, division, line, market, marketlevel, quarter, region, year. The "Selected" dimensions are: month, product, district. The "Options" section has "User Visible" and "Install" checked. The "Storage" section has "Permanent" selected. The "Width" field is empty. The "Sparse Dimensions" section shows "month, sparse <product, district>". The "Database" field is set to "DEMO".

Define a Variable

Name:

Close **Define**

General **Labels** **Format**

Data Type:

Dimensions

Available:

- choice
- division
- line
- market
- marketlevel
- quarter
- region
- year

Selected:

- month
- product
- district

Options

☒ User Visible

☒ Install

Storage

☐ Temporary

☒ Permanent

☐ In-Place

Width:

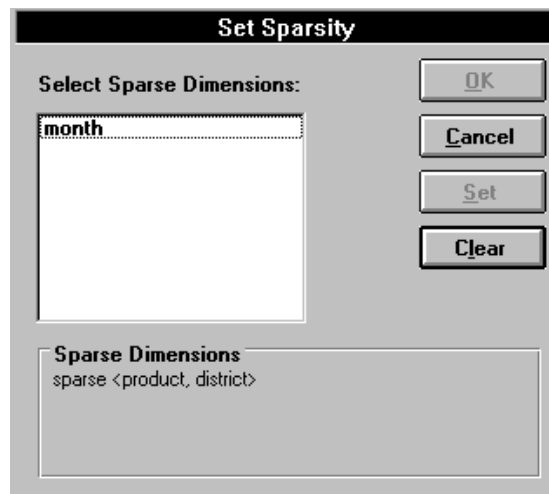
Sparse Dimensions

month, sparse <product, district>

Set Sparsity...

Database:

If you choose Set Sparsity again, notice how this information is shown here, as well. The Sparse Dimensions box displays the same “sparse <product, district>” information.



The "Set Sparsity" dialog box is shown. The "Select Sparse Dimensions:" section has a list box containing "month". The "Sparse Dimensions" section shows "sparse <product, district>". The "OK", "Cancel", "Set", and "Clear" buttons are visible.

Set Sparsity

Select Sparse Dimensions:

OK **Cancel** **Set** **Clear**

Sparse Dimensions

sparse <product, district>

The MONTH dimension is the only dimension that can be selected, because you have already set sparsity on the PRODUCT and DISTRICT dimensions.

When you define a variable in which you have set sparsity, Express automatically creates an internal structure called an unnamed composite to help process the sparse variable efficiently. A *composite* is a list of dimension value combinations that provides an index into one or more sparse variables.

For example, if you sell products in some but not all districts, your data will be sparse along the PRODUCT and DISTRICT dimensions. Therefore, when you set sparsity on the PRODUCT and DISTRICT dimensions, Express creates a composite so that dimension values are created only for those values that have data (e.g., < 'SPORTSWEAR' 'BOSTON' >).

Tip: By default, composites are created using the BTREE method of storage. Use the SPARSEINDEX option in the Express Command window to control which method of storage is used by a new composite. Acceptable values are HASH and BTREE. See the entry for SPARSEINDEX in the Express Language Help system.

The following sections describe how you can define:

- Two or more variables that share the same sparse dimensions
- A variable with more than one composite
- A variable with a single-dimension composite

If you have used a version prior to Express Server 6.0, you may have defined conjoint dimensions in your Express database. You can convert conjoint dimensions to composites, and vice versa. See Appendix B for more information.

Sharing Sparse Dimensions

When you define two or more variables with the same sparse dimensions in the same order, they share the same composite. Unless the variables differ significantly in their sparsity pattern, this is the most efficient approach.

For example, suppose you have already defined a SALES variable that has PRODUCT, DISTRICT, and MONTH dimensions. Sparsity has been set on the PRODUCT and DISTRICT dimensions. Suppose you plan to define a UNIT_SALES variable with the same dimensions as SALES. You know that UNIT_SALES will be sparse along the same dimensions as SALES, and will have the same sparsity pattern as SALES. For example, neither SALES nor UNIT_SALES will have values for GRITS in BOSTON.

►► To define variables that share the same composite, enter the dimensions in the same order, and set the same sparsity.

For example, when you define UNIT_SALES, you would select the MONTH, PRODUCT, and DISTRICT dimensions in that order (so that MONTH is the fastest-varying dimension and DISTRICT is the slowest-varying dimension). You would then set sparsity on the PRODUCT and DISTRICT dimensions. SALES and UNIT_SALES now share the same composite.

Define a Variable

Name:

General **Labels** **Format**

Data Type:

Dimensions

Available: choice, division, line, market, marketlevel, quarter, region, year

Selected: month, product, district

Options

☒ User Visible ☒ Install

Storage

☐ Temporary ☒ Permanent ☐ In-Place

Width:

Sparse Dimensions

month, sparse <product, district>

Database:

Note: In some cases, you may have two variables that are sparse along the same dimensions, but have a significantly different pattern of sparsity. To handle sparsity more efficiently in this situation, you may create a named composite to use when you define the second variable. See the section “Creating Named Composites” later in this chapter.

Multiple Composites

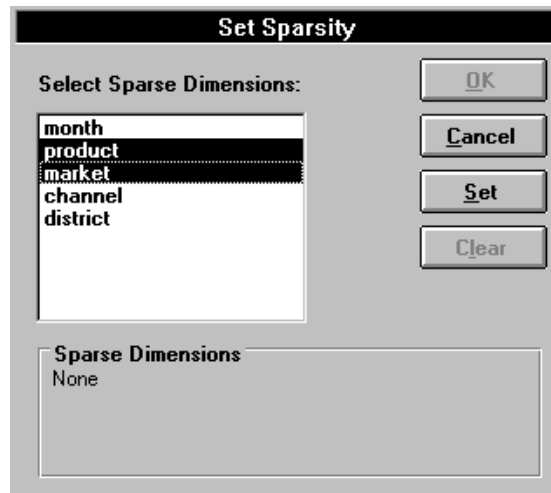
You may define a variable that has multiple composites. In other words, you set sparsity for more than one combination of sparse dimensions.

Suppose you define a UNITS variable with MONTH, PRODUCT, MARKET, CHANNEL and DISTRICT dimensions. You know that the variable will be sparse along all the dimensions but MONTH. You could define the variable with one composite by selecting PRODUCT, MARKET, CHANNEL and DISTRICT, then setting sparsity. However, if you already have variables that are sparse along PRODUCT and MARKET, there is already a composite of PRODUCT and MARKET in your database. If this variable has a sparsity pattern along PRODUCT and MARKET that is similar to those other variables, it is more efficient to use the composite that already exists. To do this, you set sparsity on PRODUCT and MARKET, then set sparsity separately on CHANNEL and DISTRICT.



To define a variable with more than one composite:

1. In the Define a Variable dialog box, specify the variable name, data type, and select the dimensions MONTH, PRODUCT, MARKET, CHANNEL, and DISTRICT in that order.
2. Choose **Set Sparsity**.
3. In the Set Sparsity dialog box, hold down the CTRL key and select PRODUCT and MARKET.



4. Choose **Set**. The sparsity you set is displayed in the Sparse Dimensions box.
5. Hold down the CTRL key and select CHANNEL and DISTRICT.



6. Choose **Set**. The sparsity you set on both pairs of dimensions is displayed in the Sparse Dimensions box.
7. Choose **OK**.
Notice that the sparsity you have selected is now displayed in the Sparse Dimensions box.

Define a Variable

Name:

General **Labels** **Format**

Data Type:

Dimensions

Available: choice, division, line, marketlevel, quarter, region, year

Selected: month, product, market, channel, district

Options

☒ User Visible ☒ Install

Storage

☐ Temporary ☒ Permanent ☐ In-Place

Width:

Sparse Dimensions

month, sparse <product, market>, sparse <channel, district>

Database:

8. Choose **Define** to define the variable.

The variable now has two composites: one composite for PRODUCT and MARKET, and one composite for CHANNEL and DISTRICT. Express automatically creates the composite for CHANNEL and DISTRICT. Since a composite already existed for PRODUCT and MARKET, the variable uses that existing composite.

Single Sparse Dimensions

It is often useful to define a variable with a single sparse dimension. In other words, when you define a variable, you set sparsity on just one dimension. When you do this, Express creates a single-dimension composite. This is a good idea when a variable will share the same dimensions as some other variables, but for a particular single dimension, the variable will only have data for some of that dimension's values.

Suppose you have already defined a variable called ACTUAL with the dimensions MONTH, ORG, and LINE. ACTUAL has no NA values. You need to define a variable called BUDGET, which requires much less detail

than ACTUAL. For example, BUDGET only needs 10 percent of the LINE dimension values, while ACTUAL needs all of them. If you define BUDGET without setting sparsity, all of the LINE dimension values are present for every MONTH and ORG, but 90 percent of the LINE dimension cells will have NA values.

To handle sparse data in this case, set sparsity on the LINE dimension only when you define BUDGET.

Define a Variable

Name:

General **Labels** **Format**

Data Type:

Dimensions

Available: channel, choice, district, division, market, marketlevel, product, quarter, region

Selected: month, org, line

Options

☒ User Visible
☒ Install

Storage

☐ Temporary
☒ Permanent
☐ In-Place

Width:

Sparse Dimensions

month, org, sparse <line>

Database:

Handling Sparse Data with Express Commands

In some cases, you may need to take advantage of Express commands to handle sparse data even more efficiently. For example, if you have two variables that are sparse along the same dimensions but have a significantly

different pattern of sparsity, it is more efficient to create a named composite that you will use when defining the second variable.

The following sections describe how to:

- Create a sparse variable with an Express command
- Create a named composite
- Name, copy, and get information about composites
- Assign data to sparse variables
- Delete composite dimension values
- Perform miscellaneous tasks with sparse variables and composites

Creating a Sparse Variable with a Command

Instead of using dialog boxes to define sparse variables, you can use Express commands, which can give you greater flexibility in handling sparse data.

When you define a sparse variable, Express automatically creates an *unnamed composite*, if one does not already exist. Regardless of whether you use a dialog box or an Express command to define a sparse variable, Express creates a composite the first time you assign sparsity for a particular combination and order of dimensions.

Administrator does not display the composite, and you do not need to do anything further. In other words, Express does all the work.

If you prefer, you can define a sparse variable by using the `DEFINE` command.



To create a sparse variable, use the `DEFINE` command with the `SPARSE` keyword. Express will automatically create an unnamed composite for you, if one does not already exist.

The syntax for defining a sparse variable with the `DEFINE` command is as follows.

```
DEFINE name datatype SPARSE <dims>
```

The syntax is explained below.

- *name* is the name of your sparse variable.
- *datatype* is the data type of your sparse variable.
- *SPARSE* indicates that the following dimensions are sparse.
- *dims* is the list of dimensions of your sparse dimension.

When you use a *SPARSE* phrase around a list of dimensions, you are indicating to Express that the variable is sparse along those dimensions. Express automatically creates an unnamed composite for those dimensions to process the variable efficiently. If you have already defined a variable with the same sparse dimensions in the same order, this variable will share the already existing unnamed composite. (For instructions on how to use a named composite in a variable definition, see the section entitled “Creating and Using Named Composite,” below.)

The following example creates a variable called *SALES* that is dimensioned by *MONTH* and a composite of the two sparse dimensions, *PRODUCT* and *MARKET*.

```
define sales integer <month sparse<product market>>
```

If you define more than one sparse variable with the same sparse dimensions listed in the same order, they share the same unnamed composite.

Creating and Using Named Composites

When you have two variables that are sparse along the same dimensions but have a significantly different pattern of sparsity, it is more efficient to create a *named composite* to use when defining the second variable. For example, two variables may be sparse along the *PRODUCT* and *DISTRICT* dimensions, but there may be many dimension value combinations where one variable has data and the other does not.

For example, suppose you have two variables named *SALES* and *PROJECTED_SALES*. These variables could have a significant difference in sparsity for many reasons. Some products and districts may have been discontinued, other products and districts may be added in the future, or

some products will no longer be sold in certain districts or will be offered for the first time in certain districts.

You refer to a named composite by the name you gave it in its definition.

A named and an unnamed composite with identical base dimensions are different composites, and they can have different dimension value combinations.



To create a variable that uses a named composite:

1. Use the DEFINE command to create a named composite. Use the following syntax.

```
DEFINE name COMPOSITE <dims>
```

In this syntax, *dims* is the list of one or more dimensions in the composite.

2. Use the DEFINE command to create a variable with the named composite. Include the named composite in the variable's dimension list. Use the following syntax.

```
DEFINE variable datatype <dims>
```

In the above syntax, *dims* is the list of dimensions of your sparse variable. In the dimension list, you use the following syntax around the dimensions that are part of the named composite.

```
composite-name<dims>
```

Using a named composite in the variable's dimension list tells Express that those dimensions in the named composite are sparse dimensions on this variable, and that this composite will only be shared with other variables using the same named composite.

The following is an example of creating a named composite called PRODUCT_MARKET and then creating a sparse variable that is dimensioned by PRODUCT_MARKET and TIME.

```
define product_market composite <product market>
define expense decimal <time product_market < >>
```

Note: Because the only dimensions that can appear in the angle brackets after a named composite are the dimensions in that named composite, you can omit the names of the dimensions and type empty angle brackets, as shown in the above example. Express will fill in the names of those sparse dimensions for you.



For more information about named composites, see the `DEFINE COMPOSITE` command in the Express Language Help system.

Naming, Copying, and Getting Information About Composites

It is possible to perform various dictionary operations on composites to change or query your database. You can name composites, copy the definition of one composite to another composite, and query your database for information about named composites.

Naming Composites

You may name an unnamed composite.



To name an unnamed composite, use the `RENAME` command.

The following is an example of naming an unnamed composite. A variable has already been defined in which sparsity has been set on the `PRODUCT` and `MARKET` dimensions. This example assigns the name `PRODUCT_MARKET` to the variable's previously unnamed composite.

```
rename sparse <product market> product_market
```

You can change the name of a named composite with the `RENAME` command. For example, to change the name of `PRODUCT_MARKET` to `PRODMARK`, use the following command.

```
rename product_market prodmark
```

You can also change the name of a named composite to an unnamed composite if the composite has no properties or permission restrictions and

if there is at least one object dimensioned by it. In addition, there cannot be more than one unnamed composite with the same dimensions in the same order in the database.

Use the following syntax to change a named composite to an unnamed composite.

```
RENAME composite-name SPARSE
```

Copying Composite Definitions



To copy the definition of a named composite, use the COPYDFN command. See the entry for COPYDFN in the Express Language Help system for information about using COPYDFN.

You can also use the COPYDFN command to copy the definition of sparse variables or any objects in your database.

Getting Information About Composites



To get information about a named composite, use the OBJ and DATABASE functions.

The COMPOSITE “choice” argument of the DATABASE function returns a list of all named composites in a database.

The OBJ function returns information about database objects. You can use the OBJ function to get information about named composites and to get information on variables with composites. The following tables show a subset of OBJ choices that you might want to use.

Choice	Data Type of Result	Information Returned
DATA	TEXT	For a composite, the names of the dimensions in the composite as a multiline text value.
DFNDIMS	TEXT	For a composite or a variable, a multiline text value containing the name(s) of the dimensions and composites in the dimension list used to define an object.

DIMS	TEXT	For a composite or a variable, a multiline text value containing the name(s) of an object's dimensions. For composites, it returns a multiline text value listing the base dimensions of the composite. For variables, it returns a multiline text value containing the name(s) of the dimensions of the variable.
ISBY dim	BOOLEAN	For a composite, specifies whether or not an object was defined with the composite you specify in "dim."
NUMDFNDIMS	INTEGER	For a composite or a variable, the number of dimensions or composites in the dimension list used to define an object. For this count, each composite counts as one, and the dimensions within the composite's dimension list are not counted.
NUMDIMS	INTEGER	For a composite or a variable, the number of dimensions of a dimensioned object. For all types of dimensions, NUMDIMS returns 1. For a composite, it returns the number of base dimensions.
SPARSE	TEXT	For a composite or a variable, a multiline text value listing the named or unnamed composites used in the definition of an object.
TYPE	TEXT	For a composite, COMPOSITE.



See DATABASE and OBJ in the Express Language Help system for detailed information about these functions.

Assigning Data to Sparse Variables

You may assign data to sparse variables in a number of ways. You can edit variables or import data, as described earlier in this manual. You can use the = command to assign the value of an expression to your sparse variable. You can use the FILEREAD command to read data from a file into your

sparse variable. You can use the SQL FETCH command to fetch data from a relational database into your sparse variable.

No matter what method you use to assign data to a sparse variable, Express will automatically add composite values as needed to create any missing target cells that are being assigned non-NA values.

If the sparse dimensions in your sparse variable are *not* last in the dimension list for the variable, the data storage for your variable could become inefficient. If this is the case, see the section “Using CHGDFN SEGWIDTH for Execution Efficiency” before you assign data to your sparse variable. If the sparse dimensions are last in the dimension list, the data storage for your variable should be reasonably efficient without any work on your part.

The following sections describe alternative ways to assign data to sparse variables.

The = Command

When you use the = command to assign data to a variable, the variable appears on the left side of the = sign, and is called the *target variable*. The data you are assigning to the variable appears on the right side of the = sign, and is called the *source expression*.

When you assign data to a sparse variable with the = command, Express automatically adds composite values as needed to create new data cells for non-NA data. For example, suppose you assign data to the SPARSE_SALES variable. Express attempts to assign values from the source expression into the target variable for all the combinations of products and markets that are in status. If there is no existing data cell in the target variable for a particular product\market combination, and the source expression has data for that combination, Express adds the missing dimension value combination to the composite, thus creating the missing cell. It then sets the cell to the value in the source expression. The following example assigns data to the sparse variable SPARSE_SALES from the SALES variable, automatically adding values to the composite used by SPARSE_SALES as needed.

```
sparse_sales = sales
```

Express always assigns data correctly, regardless of the sparsity of the target variable or the source expression. For example, the target variable may be a sparse variable while the source is a regular variable. Or the source may be a sparse variable and the target may be a regular variable. Or both target and source may be sparse variables; they could have the same composite, or different composites. Express will handle each case appropriately.

If the source expression is sparse, and Express tries to find a value for a dimension value combination that does not exist in the source expression, it will consider that the source expression has an NA at that dimension value combination.

When assigning data to sparse variables, Express evaluates the source expression for every combination of the dimension values in status for the target variable, including combinations of the sparse dimensions for which the target variable currently has no cells. If the source expression is not NA for those combinations where the target currently has no cells, Express will create new cells and assign the data into them.

Express gives you the ability to specify a different evaluation behavior. You can alter the default evaluation behavior of the = command so that Express evaluates the source expression only for those combinations of the dimension values in status for which the target variable currently has cells. Since the composite of the sparse dimension is how Express keeps track of which combinations of the sparse dimensions have data cells, you use the following syntax to specify this different evaluation behavior.

```
name = expression ACROSS composite
```

The syntax is explained below.

- *name* is the sparse variable. It is the target where data will be assigned and stored.
- *expression* is the source expression that holds the data that will be assigned to the target variable.
- *ACROSS* is the keyword that indicates that you want to alter the default evaluation behavior and cause Express to evaluate the composite of the target variable.

- *composite* is the composite for the sparse dimensions on the target variable.

If the variable was defined with a named composite, specify the name of the composite. If the variable was defined with an unnamed composite, use the following syntax to refer to the unnamed composite.

`SPARSE<dims>`

In this syntax, *dims* is the list of the dimensions in the composite. These are the dimensions you indicated as sparse when you defined the variable, either by using the Set Sparsity dialog box, or by using the DEFINE command with the SPARSE keyword.

For example, to have Express assign data from SALES only into existing data cells of SPARSE_SALES, whose associated dimension values are in status, use the following command.

```
sparse_sales = sales across sparse<product market>
```

The ACROSS keyword is particularly helpful if the source expression is a single value. If there are no limits on the dimensions of SPARSE_SALES, then an assignment command like the following will create cells for every combination of dimension values, because there are no cases where the source expression is NA.

```
sparse_sales = 0
```

This defeats the purpose of a sparse variable.

In contrast, the following command will set only existing cells of SPARSE_SALES to 0.

```
sparse_sales = 0 across sparse<product market>
```

Another case where the ACROSS keyword is particularly useful is to increase the efficiency of an assignment statement like the following.

```
sparse_sales = sparse_sales * 1.1
```

Without an ACROSS keyword, Express will unnecessarily evaluate every combination of the dimension values in status. For every dimension value

combination for which SPARSE_SALES does not have a data cell, the source expression that is based on SPARSE_SALES will evaluate to NA. No assignment will happen for those dimension value combinations, because Express does not add new data cells when the source expression is NA. In contrast, the following command evaluates the source expression only for those combinations of the dimension values in status for which SPARSE_SALES has existing data cells.

```
sparse_sales = sparse_sales * 1.1 across  
sparse<product market>
```

You can also use the = command with a qualified data reference. The following example will assign the value 10200 to the data cell specified in the qualified data reference. If the composite does not already have a value for the combination 'BOSTON' and 'TENTS', this value combination will be added to the composite, thus adding the data cell.

```
sparse_sales(market 'BOSTON' product 'TENTS' month  
'JAN99') = 10200
```

The FILEREAD Command

When reading data into a sparse variable (i.e., one defined with a composite in its dimension list), FILEREAD automatically creates any missing target cells that are being assigned non-NA values. This process also adds to the composite all the dimension value combinations that correspond to those new cells.

Note that the syntax for the FILEREAD command is the same as it would be for a regular variable.

```
fileread unit append w 8 product append w 8 market w 10  
sparse_sales
```

If the sparse dimensions in your sparse variable are *not* last in the dimension list for the variable, the data storage for your variable could become inefficient.

As with any method of assigning data to sparse variables, you can use the CHGDFN command with the SEGWIDTH keyword before assigning data to make the storage more efficient. See the following section entitled “Using CHGDFN SEGWIDTH for Execution Efficiency.”

Alternatively, you can use the `FILEREAD` command to explicitly add values to the composite for your sparse variable before reading in the data. See the `FILEREAD` entry in the Express Language Help system for more information.

Using `CHGDFN SEGWIDTH` for Execution Efficiency

If you have defined a sparse variable whose sparse dimension is *not* the slowest-varying dimension, it is critical that you use the `CHGDFN` command to ensure good performance *before* that variable contains any data. You can also use this command to ensure good performance for any variable.

Use the `CHGDFN` command with the `SEGWIDTH` keyword to specify the size of a variable's segments. A *segment* is a portion of the total number of values a variable holds. The number of segments in a variable affects the performance of data loading and data accessing. Too many segments (thousands to millions) can degrade performance.

If you have a sparse variable whose sparse dimensions are *not* last in its dimension list, that variable will have a segment for every dimension value combination where Express has automatically created a new composite value as it added data. You can reduce the number of segments if you let Express know how many values you expect the composite for the variable will have once all the data has been assigned.

Use the following syntax for `CHGDFN`.

```
CHGDFN varname SEGWIDTH length-dim1 length-dim2 ...
```

Specify the segment width as the maximum number of values per segment for each dimension or composite on the variable. *Length-dim1* is the number of values for the dimension or composite in the first position of the dimension list of the variable's definition (the fastest-varying dimension or composite), *length-dim2* is the number of values for the dimension or composite in the second position in the dimension list, and so on.

If you are not sure what your segment size should be, use the maximum anticipated number of values for each dimension or composite as the length arguments to `SEGWIDTH`. Then only one segment will be created for the variable.

Suppose you have a variable SPARSE_UNITS where the sparse dimensions are not at the end of its dimension list. The following is the definition of the SPARSE_UNITS variable.

```
define sparse_units variable integer <sparse<district  
product> month>
```

Now suppose you would like this variable to have only one segment. SPARSE_UNITS's dimension list includes an unnamed composite of the DISTRICT and PRODUCT dimensions (SPARSE<DISTRICT PRODUCT>), and the MONTH dimension. If you expect eventually to have data for 90000 combinations of DISTRICT and PRODUCT, and the MONTH dimension to have 120 values, you can make the segments for SPARSE_UNITS that size by using the following command.

```
chgdfn sparse_units segwidth 90000 120
```

This CHGDFN command will affect the segment size not only for SPARSE_UNITS, but also for any other variable that has a dimension list identical to the dimension list for SPARSE_UNITS. This means that if you have a number of sparse variables with identical dimension lists, a single CHGDFN command will set up an efficient segment size for the whole group of variables.

Important: You must execute the CHGDFN command before any data is assigned to a variable in order to have just *one* data segment for that variable.

If data has already been assigned to SPARSE_UNITS, or any other variable with an identical dimension list, several segments have already been created. The CHGDFN command cannot get rid of those existing segments. It can, however, reduce the number of segments that are created *after* you execute the CHGDFN command. You can change the segment width of a variable at any time to further optimize the number of segments.

If you are an experienced Express user and your database has large variables, you may want to have more than one segment per variable. You can specify segment sizes for those variables that best suit their application. Sometimes, having more than one segment can improve performance. If

you want your variable to have more than one segment, it is generally a good idea to keep the number of segments small.

You can use OBJ(NUMSEGS) to find out if you have too many segments for objects that have a particular dimension set. For example, to find out the number of segments for the variable SALES, enter the following commands in the Express Command window.

```
consider sales
show obj(numsegs)
```

If you find that an object has too many segments, you can reduce the number of segments. See the CHGDFN command and the OBJ function in the Express Language Help system for details.

If you have previously set the segment size for a particular dimension set, you can check on the size by executing a command with the following syntax.

```
CHGDFN variable-name SEGWIDTH
```

When you use the SEGWIDTH keyword with no length arguments, Express produces an error message that includes the current *length-dim* value for each dimension in the dimension set for *variable-name*.

There are some alternative methods for ensuring efficient segment size. If you are using the FILEREAD command to assign data into your sparse variable, you can use the FILEREAD command to explicitly add values to the composite for your sparse variable before reading in the data. See the FILEREAD command in the Express Language Help system for more information. In some situations, you might want to use the MAINTAIN command to explicitly add values to the composite before assigning data. See the MAINTAIN command in the Express Language Help system for more information.

Deleting Composite Dimension Values

If you need to delete composite values, use the MAINTAIN command with the DELETE keyword. You can delete one or more specified values of the composite, or you can delete related groups of composite values.

The syntax for deleting one or more specified values from a composite is as follows.

MAINTAIN composite DELETE valuelist

Composite can be either a named or unnamed composite. To refer to a named composite, use its name. To refer to an unnamed composite, use the keyword **SPARSE** followed by the dimensions of the composite enclosed by angle brackets. *Valuelist* is a list of composite values. You refer to an individual composite value the same way you refer to values of a conjoint dimension: by enclosing the values of the composite's base dimensions in angle brackets.

For example, suppose you have an unnamed composite with the base dimensions of **PRODUCT** and **DISTRICT**. To delete the value `<'SNOWSHOES' 'ATLANTA'>` from that composite, you would use the following command.

```
maintain sparse<product district> delete <'SNOWSHOES'
'ATLANTA'>
```

You can also use the **MAINTAIN** command to delete related groups of composite values by using the following syntax.

MAINTAIN composite DELETE basedim basevaluelist

Basedim is one of the base dimensions of the composite. *Basevaluelist* is a list of one or more values of the specified base dimension.

For example, suppose you no longer need any of the data related to the product **SNOWSHOES** in any of the sparse variables sharing the unnamed composite with the base dimensions of **PRODUCT** and **DISTRICT**. To delete all the composite values that contain **SNOWSHOES** as their product dimension value, use the following command.

```
maintain sparse<product district> delete product
'SNOWSHOES'
```

The above command will remove all the composite values that contain **SNOWSHOES**. It does not remove **SNOWSHOES** from the **PRODUCT** dimension.

You can delete a dimension value from one of the base dimensions of a composite by using the `MAINTAIN` command directly on that base dimension. This will also delete values associated with that dimension value from the composite. For example, suppose your company drops `SNOWSHOES` entirely from its product line. Therefore, you need to remove `SNOWSHOES` from the `PRODUCT` dimension as well as from the composite. To do this, use the `MAINTAIN` command directly on the `PRODUCT` dimension, as shown in the following command.

```
maintain product delete 'SNOWSHOES'
```

The above command not only removes `SNOWSHOES` from the `PRODUCT` dimension, but it also automatically removes any composite values associated with `SNOWSHOES` from any named or unnamed composites that have `PRODUCT` as a base dimension.

Performing Miscellaneous Tasks

Variables and other objects that are defined with composites generally behave like variables dimensioned by simple dimensions.

The following sections describe miscellaneous tasks you may need to perform and additional information you may need to know when using composites or sparse variables.

Limiting Dimensions

You use the `LIMIT` command to set status for the dimensions of a sparse variable in the same way you would if the variable were not sparse.

The following example sets the status of the `PRODUCT` and `MARKET` dimensions prior to looking at the `SPARSE_SALES` variable, which is sparse along `PRODUCT` and `MARKET`.

```
limit product to 'CANOES' 'SPORTSWEAR'  
limit market to 'BOSTON' 'ATLANTA'
```

Composites are not dimensions, and therefore do not have any independent status. The values of a composite that are “in status” are determined by the status of the base dimensions of the composite. In general, when Express functions and commands deal with objects defined with composites, the

default behavior is to treat those objects as if no SPARSE keyword or named composite had been used when the object was defined.

Displaying Data in a Report

When you use the REPORT command with a sparse variable, Express's default behavior is to evaluate all the combinations of the values of the composite's base dimensions that are in status. Any combinations that do not exist in the composite display NA for their associated data.

For example, suppose you want to look at SPARSE_SALES for sales in Boston from January 95 to March 1995. First, use LIMIT commands to limit the status of DISTRICT to 'BOSTON' and the status of MONTH to 'JAN95,' 'FEB95,' and 'MAR95.' The composite on DISTRICT and PRODUCT only has values for <'CANOES' 'BOSTON'> and <'SPORTSWEAR' 'BOSTON'>.

Use the following command to produce a report for SPARSE_SALES.

```
report SPARSE_SALES
```

The above command will produce the following output.

DISTRICT: BOSTON

PRODUCT	SPARSE_SALES		
	MONTH		
	JAN95	FEB95	MAR95
TENTS	NA	NA	NA
CANOES	66,014	76,084	91,748
RACQUETS	NA	NA	NA
SPORTSWEAR	53,195	58,913	62,798
FOOTWEAR	NA	NA	NA

You can override this default REPORT behavior by specifying the composite on your sparse variable in an ACROSS, DOWN, or GROUP phrase. Besides changing the format of a report, specifying a composite in an ACROSS, DOWN or GROUP phrase in a REPORT command changes what is shown in the report. When you use such a phrase, the report will show only those combinations of dimension values which both exist in the composite and are in status.

For example, to report the SPARSE_SALES variable broken out by its composite, with the values of the composite going down the report, use the following command.

```
report down sparse<product district> SPARSE_SALES
```

Because the composite only has values for the <'CANOES' 'BOSTON'> and <'SPORTSWEAR' 'BOSTON'> combinations, the above command would produce the following output.

		-----SPARSE_SALES-----		
		-----MONTH-----		
PRODUCT	DISTRICT	JAN95	FEB95	MAR95
CANOES	BOSTON	66,014	76,084	91,748
SPORTSWEAR	BOSTON	53,195	58,913	62,798

See the REPORT command in the Express Language Help system for more information.

Rolling Up Data for Hierarchical Dimensions

The ROLLUP command calculates totals for a hierarchy of dimension values where each level of the hierarchical dimension is an aggregation of the level below it. If you are rolling up a sparse variable, a hierarchical dimension can be a sparse dimension of the variable. Express will automatically create any missing cells of the sparse variable needed to store the calculated totals.

See the ROLLUP command in the Express Language Help system for more information. Chapter 4 gives an overview of creating and using hierarchical dimensions, including information about using the ROLLUP command.

Models

You can use a sparse variable as a solution variable when solving a model. Express will automatically create any missing cells in the sparse variable needed to store the result of equations executed when solving the model.

See the MODEL command in the Express Language Help system for more information on models. Also, refer to *Express Language Programming Guide* for information about creating and using models.

Keeping Your Database Efficient

Frequent maintenance of a dimension, such as adding, deleting, or reordering its values, can affect the efficiency of your application. This is because when you maintain a dimension, the *physical order* of its values (the order in which they are actually stored) can become different from the *logical order* (the order in which the values appear when you use your database), making data retrieval less efficient. In addition, when you delete values from a dimension, Express Server marks those values as deleted, but the space they occupied is unusable. Although the dimension values no longer appear in your dimension, Express Server cannot use this “dead” space for other purposes.

This eventual “disorganization” of a dimension that receives frequent maintenance can slow down processing in your application because it affects not only the dimension itself, but also any object that depends on it: composites, variables, or relations that are dimensioned by it or related to it.

The next section explains how to eliminate the dead space created by dimension maintenance and bring the physical order and logical order of a dimension back into correspondence.

Making the Database File Smaller

If you want to use disk space efficiently, you can consider rebuilding your database. When you rebuild the database, your database takes up less space on your disk.

While rebuilding a database will reduce the disk space required, the process can take a long time if your database is large. In addition, compiled program code saved in your database will be discarded during the process. This means that after you rebuild the database, you may want to recompile all the programs in the database (easily done with the ALLCOMPILE command).

The following procedure describes how to rebuild a database named MARKET. Keep in mind that for a large database, this procedure can take a considerable amount of time to perform.

**To rebuild a database:**

1. From the Tools menu, choose **Express Command** to display the Express Command dialog box.
2. Set the status of all dimensions to ALL. To do so, enter and execute the following commands in the Express Command dialog box.

```
database attach market first  
limit name to all  
allstat
```

3. Export all the database objects to an Express Interchange Format (EIF) file. From the Tools menu, choose **Export EIF** to display the Export an EIF File dialog box, in which you specify a file name for the exported data (i.e., MARKET.EIF), then specify the objects you want to export.
4. Rename the file for the database you are rebuilding. You probably will not need the file again, but you should not delete it until you know that you have successfully imported everything in it. By renaming the database, you can keep the old database as long as you want, while using the original database name for the rebuilt database.

Since you cannot rename a database in Administrator, you must rename the database at the system level. (Your system administrator might have to do this.) For example, if your database is named MARKET, you can rename it OLDMARK.

5. Create a new database with the original name of the database you just renamed. From the File menu, choose **New Database** to display the Create a New Express Database dialog box, and enter the original database name (i.e., MARKET).
6. In the new MARKET database, import all the database objects you previously exported to an EIF file. From the Tools menu, choose **Import EIF** to display the Import an EIF File dialog box, in which you specify the file from which you want to read data and definitions (i.e., MARKET.EIF), then specify the source database objects you want to import.

You should examine your MARKET database to make sure all the objects and data are there. From the View menu, choose **All Objects**. This displays all of the objects in your database in the Database Browser. Then you can

delete the old database (OLDMARK) or you could copy it to a tape or CD and keep it as a backup.

As noted above, IMPORT and EXPORT do not transfer compiled programs, just program text. For efficiency, you can compile all the programs with the ALLCOMPILE command right after you import the database.

Tip: Because EIF files are smaller than database files, they are an excellent way to keep backups of Express Server databases.

Conserving Memory While Running Express Server

Conserving memory is only a concern if a server is running other programs in addition to Express Server.

Your system administrator decides how much memory each job on the Express Server system needs. It may take some fine-tuning to arrive at the best memory allocation for that system. You do not want to use too much memory because that would take system resources away from other jobs. On the other hand, if you have too little memory, you will not have enough room for all the definitions and data contained in a large database. Express Server will have to page data and code in and out frequently, increasing the number of times it has to read the disk. Too much paging slows Express Server down because it takes longer to retrieve data.

The suggestions in this section are only for cases where databases are very large or application code is very complex.

Express Server divides its available memory into two sections:

- Permanent memory: for object definitions
- Temporary memory: for program code, data, and computations

When you start an Express Server session, Express Server reads the definition of each object you use and stores it in permanent memory. As you use more objects, the amount of memory that is used increases. Its size

never decreases; memory can be reused but not returned to the temporary-memory region. Temporary memory is used for processing commands and storing current data and program code. Processing complex commands and recursive formulas requires more memory. If you run out of memory while processing a complex command or expression, you can separate it into several simpler commands.

If you see messages indicating that the system is out of memory, you can try the following suggestions. The first three recommendations are decisions you make when you design your database. They really are the same as the recommendations for making a small database file. The last three have to do with the way you write application programs. You do not necessarily have to worry about them unless you see warning messages.

To minimize the amount of memory needed to run Express Server:

- Use temporary variables. Their definitions take less space.
- Use composites by setting sparsity when you define variables. Composites reduce the size of variables with sparse data both on disk and in memory.
- Do not define more objects than necessary. Each object definition takes up memory. Definitions for temporary variables and relations take up less room than other definitions.

If you write programs, you can also:

- Minimize the use of recursive formulas. Each iteration of the formula takes up memory.
- Break up deeply nested character string expressions.
- Write a program that uses the ROW command, but reduce its complexity. Processing the ROW command takes a lot of memory.

Some of these recommendations seem contrary. For example, composites create three new objects: a new dimension and a relation to each of the original dimensions. However, the savings in the number of data cells can outweigh the size of the extra definitions. You have to select priorities that are applicable to your system and optimize your database for those conditions.

Saving a Database in Two or More Files

Ordinarily, an Express Server database is contained in a single file. However, a database can be a *multifile database*; that is, it can span several files. In addition, the files can be located on separate qualifying locations. A *qualifying location* is a disk, volume, file system, pathway, or device-and-directory (depending on your system) that has some space for more files. This way, you can create a database that is larger than any single location.

A multifile database is spread over a main file and extension files. Express Server creates the *main file* when it first creates a database. As you add and configure database objects, Express Server adds to the main file until it reaches the maximum size you specified for it with the FILESIZE argument (shown below) or until the current disk becomes full. When this happens, Express creates a database *extension file*. You can identify the location where Express should create any database extension files by setting the text option DBEXTENDPATH. The value of DBEXTENDPATH overrides the Express Configuration Manager setting for ExtensionFilePath. For more information, see the entries for DATABASE and DBEXTENDPATH in the Express Language Help system.

When the extension file reaches the maximum size or fills its location, Express Server creates another extension file, and so on. If the end of the list is reached and there are still qualifying locations, Express Server returns to the beginning and goes through the list again until it saves the whole database or uses up all the space everywhere in the list and signals an error.

To create a multifile database, you first create a database, then use a DATABASE FILESIZE command to specify the maximum size of the main file and any extension files. In the Express Command dialog box, you would type and execute the following commands.

```
database filesize 32m
```

Using the FILESIZE keyword indicates that you want Express Server to create an extension file when the current file reaches the specified size. If you just specify a number after the keyword, the size is evaluated in bytes.

If you specify “K,” “M,” or “G” after the number, the size is evaluated in kilobytes, megabytes, or gigabytes.

If you want to change the maximum file size, you can use the DATABASE FILESIZE command at any time. The new file size will not change the size of existing files, but it will affect the size of a file that is currently being filled and the size of all subsequent extension files.

The file name of the main database file and all extension files is the database name you specified when you created the database. For the file type of the main file, “.DB” is automatically appended to the name you specify.

For the extension files, file types that contain numbers are automatically appended. The first extension file is assigned a file type that contains the number “001” and subsequent extension files are assigned file types containing the numbers “002,” “003,” “004,” and so on. Extension file *names* must always match the main file name, and extension file *types* must never be changed.

Note: If Express Server finds even one of a database’s extension files in a read-only location, it treats the entire database as read-only. Note also that if you delete even one of a multifile database’s extension files, Express Server will no longer be able to open the database and the whole database will be inaccessible. On the other hand, if you delete a multifile database, remember to delete all of its extension files.

You can use the DATABASE function with the “pathname” choice to list all the files that make up the multifile database. For example, to list the file names and directory paths for the current database, type the following command in the Express Command window.

```
show database(pathname)
```

When you execute this command, the file and pathnames of the current database will be displayed in the Express Command window. For more information, search for the DATABASE function in the Express Language Help system.

Controlling Access to Your Database

You can take advantage of NT security features by adding settings to Administrator and client application .INI files. These settings are discussed in detail in Chapter 2.

This section describes how to create a password for a database, as well as Express commands you can use in a program to specify security.

Creating a Password for a Database

After creating the database, you can assign a password to it. To do so, first select **Express Command** from the Tools menu to display the Express Command dialog box. Use the Express DATABASE command to enter a password for the database in the dialog box. Be sure to click on the **Execute** button to execute the command you enter.

Note: If you currently have more than one attached database, any DATABASE command you execute affects only the top attached database. To ensure that the database to which you want to assign a password is the top attached database, execute a DATABASE ATTACH *name* FIRST command in the Express Command window, where *name* is the name of your database, and FIRST makes your database the current database. You can then execute a DATABASE LIST command, which displays the order in which your databases are currently attached.

You use the PASSWORD keyword with the DATABASE command, followed by the password you want to assign. For example, to assign the password “xxx,” you would enter the following.

```
database password xxx
```



For details, refer to the DATABASE command entry in the Express Language Help system.

Levels of Database Security

You can use the Express commands described in this section in Administrator or you can use them in a program. Alternatively, if you have used previous versions of Express, you may already have programs that use these commands and may need to know more about them.

You can create security applications that control access to your database by using database permission programs, the PERMITERROR option, and the PERMIT and PERMITRESET commands. You can define a PERMIT_READ or PERMIT_WRITE program that will run when the database is attached either for read-only or for read/write access. Your database permission programs can contain PERMIT commands and values for *permission conditions*. PERMIT commands assign different types of access permission to objects in your database, depending on the permission conditions that you establish. You could set up your permission conditions to test for a class of database users.

Operating System Access Rights

In some cases, you can specify read or write access to database files within the operating system. If you can establish or restrict database access through the operating system and you do not need to establish access rights to objects within the database, then you might not need to use the security mechanisms in Express Server.

Database Password

If you have established a password for a database with the DATABASE PASSWORD command, Express Server checks for that password, before attaching the database. However, if a database permission program appropriate to the attach mode (read-only or read/write) exists within the database, Express Server passes the password as an argument to the database permission program instead of validating it against the stored database password.

Database Permission Programs

In each database, you can create database permission programs called `PERMIT_READ` and `PERMIT_WRITE`. These programs must be user-defined boolean functions. When you attach a database with read-only access, Express Server executes the `PERMIT_READ` program, if it finds one in the database. When you attach a database with read/write access, Express Server executes a program called `PERMIT_WRITE`, if it finds one in the database.

When a database permission program is executed, it must return `YES` in order for the database to be attached. Database permission programs run before an `AUTOGO` program.

Within the database permission programs, you can use `PERMIT` commands to establish access conditions for objects in the database. You can also specify values for the variables that you will use in the permission conditions for `PERMIT` commands in the database. These variables might be passwords or user IDs. You can protect the `PERMIT_READ` and `PERMIT_WRITE` programs themselves with `PERMIT` commands.



See the following section for more information on the `PERMIT_READ` and `PERMIT_WRITE` programs. Also, refer to the entry for the `PERMIT` command in the Express Language Help system.

The PERMIT Command

The `PERMIT` command lets you control access to database objects. With `PERMIT`, you can grant or deny read-only and read/write access for objects and for values of dimensions and dimensioned objects. You can also use `PERMIT` to grant or deny permission to maintain dimensions and to change permission for objects. You can apply up to four `PERMIT` commands to an object: one for read, write, maintain, and permit permission. Before executing a `PERMIT` command, you must use the `CONSIDER` command on that object.

You can use the PERMITRESET command to cause Express Server to reevaluate object permission. You can use the PERMITERROR option to control how Express Server handles attempted violations of permission.

PERMIT can use permission conditions based on the value of SYSINFO(USER) or SYSINFO(GROUP). SYSINFO(USER) returns the NT identity under which the session is running. SYSINFO(GROUP) returns the groups to which the NT identity belongs. You can use the values returned by SYSINFO(USER) and SYSINFO(GROUP) with Express PERMIT commands to create security mechanisms within Express.



For complete descriptions of PERMIT, PERMITRESET, PERMITERROR, and SYSINFO, see the Express Language Help system.

Establishing Permission Conditions

The conditions for granting read, write, maintain, or permit permission to a database object consist of one or more boolean expressions. These expressions can be dimensioned or undimensioned.

The permission condition consists of a single boolean value in the following cases.

- When you are assigning permission to an object without dimensionality
- When you are assigning permission to *all* the values of a dimensioned object
- Whenever you are executing PERMIT MAINTAIN or PERMIT PERMIT commands

For example, the following commands allow the program MYPROG to be available for editing as well as for execution as long as the boolean variable MYBOOL is set to YES. When MYBOOL is NO, MYPROG can be executed but not edited.

```
define mybool variable boolean
define myprogprogram
permit write when mybool eq yes
```

The commands that follow cause the PRODUCT dimension to be available for maintenance as long as MYBOOL is set to YES. When MYBOOL is NO, you cannot perform maintenance on the PRODUCT dimension.

```
consider product
permit maintain when mybool eq yes
```

With dimensioned permission conditions, you can assign read or write permission to individual values and to slices of values within dimensioned objects.

For a SALES variable dimensioned by MONTH, PRODUCT, and DISTRICT, you might have three dimensioned permission conditions in the form of three variables as shown in the following example.

MONTH_BOOL	<MONTH>	PROD_BOOL	<PRODUCT>	DISTRICT_BOOL	<DISTRICT>
JAN95	NO	TENTS	YES	BOSTON	NO
FEB95	YES	CANOES	YES	ATLANTA	NO
MAR95	NO	RACQUETS	NO	CHICAGO	YES
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.

If the YES values in this example are the only YES values in the permission conditions, the following PERMIT command provides read/write access to SALES data for TENTS and CANOES sold in CHICAGO in FEB95. In the absence of a PERMIT READ command for SALES, Express Server provides read-only permission for all the other values of SALES.

```
permit write when district_bool by prod_bool by month_bool
```

If you want to prevent access to one particular sales figure, say for racquets in Boston in March of 1997, you can create a boolean variable and use it in a PERMIT command as shown below. SALES_BOOL is a boolean variable dimensioned by MONTH, PRODUCT, and DISTRICT.

```
sales_bool = yes
limit month to 'MAR97'
limit product to 'RACQUETS'
limit district to 'BOSTON'
sales_bool = no
consider sales
permit read when sales_bool
```

Assigning Permission to a Dimensioned Variable

The PERMIT command is designed to handle certain typical cases with as little coding as possible. Of the five cases illustrated in the following examples, four can be specified with a single PERMIT command, and one requires two PERMIT commands. This concise syntax is possible because of the permission defaults associated with the PERMIT command.

When you target a dimensioned variable for read or write permission, a single PERMIT command can grant or deny the other permission by default. Unless you explicitly override this default behavior with another PERMIT command, the following rules apply:

1. When you deny read permission, write permission is denied for those values by default.
2. When you grant read permission, write permission is granted for those values by default.
3. When you grant *or* deny write permission, read permission is granted for *all* values by default.

Note: It is possible to grant write permission to a slice of a dimensioned variable to which you have not granted read permission. This means that these values can be modified but not viewed. To create this situation, you need to specify both a PERMIT READ and a PERMIT WRITE command for the variable, and some or all of the values targeted for write permission must not also be targeted for read permission.

The typical permission cases, shown in Figures 9.6 through 9.10, use a two-dimensional variable called MYVAR. The figures identify the areas of the data where the permission conditions apply. Each case identifies the appropriate syntax of the PERMIT command.

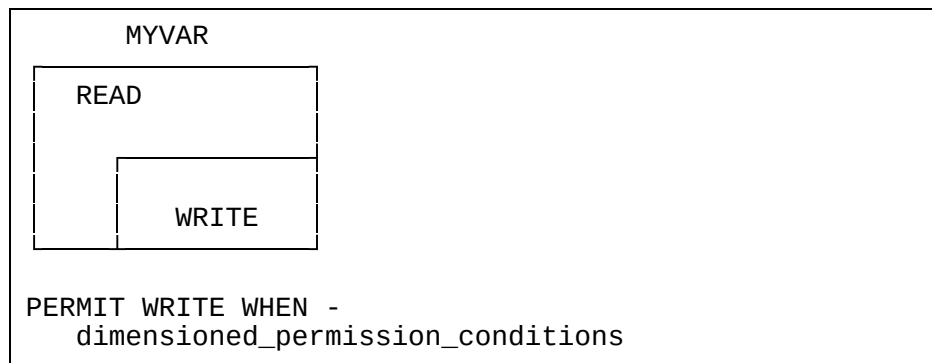


Figure 9.6 Full read permission and restricted write permission.

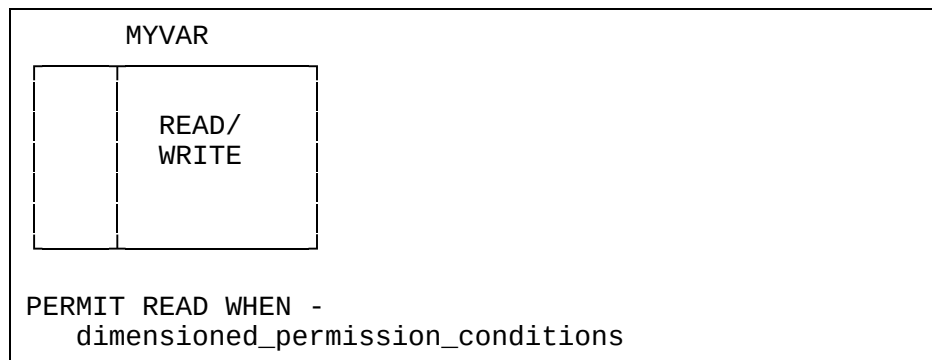


Figure 9.7 Restricted read and write permission where the areas for read and write permission are the same.

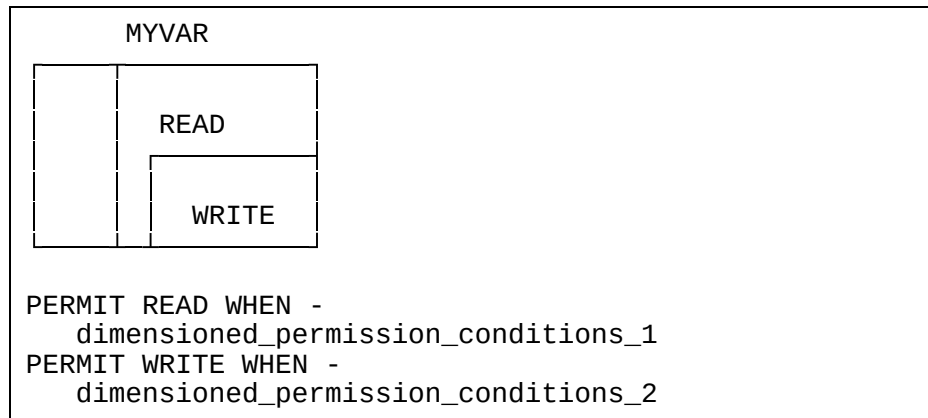


Figure 9.8 Restricted read permission with a subset of that area having write permission.

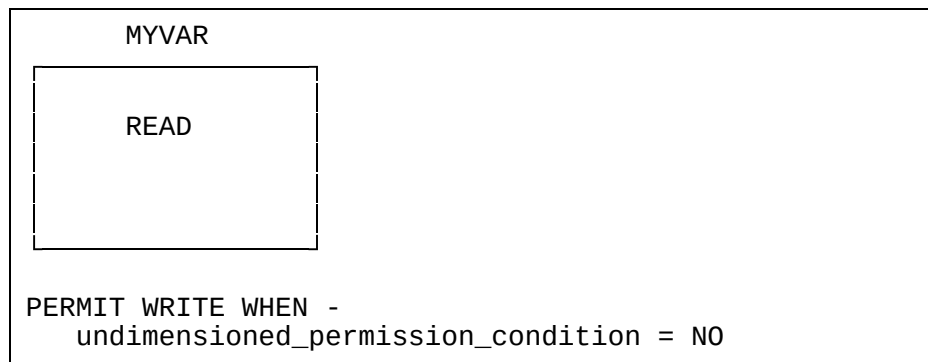


Figure 9.9 Read-only permission for the whole object.

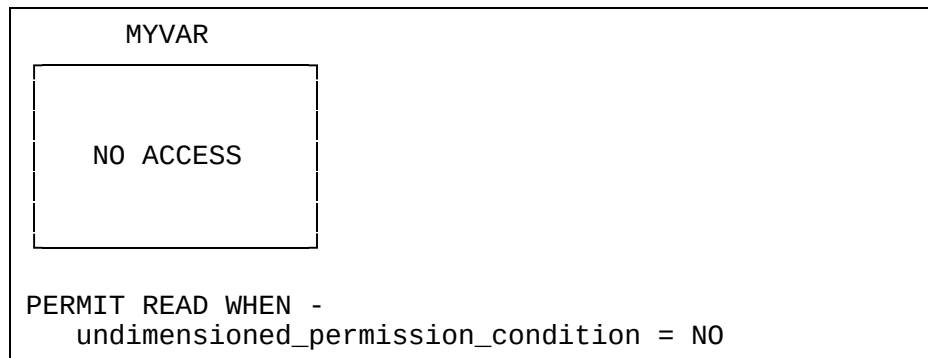


Figure 9.10 Deny permission to access the object entirely.

Assigning Permission to a Dimension

You can grant read or write permission to dimensions. You can use a dimensioned permission condition to target dimension values, or you can use a single-cell permission condition to target the whole dimension.

Dimension values that satisfy the permission conditions for a PERMIT READ command constitute the *default status* of a dimension. Therefore, the permission shown below effectively removes all MONTH values for the years prior to 1996 from your database.

```
describe month
```

```
DEFINE MONTH DIMENSION MONTH
LD MONTH
PERMIT READ WHEN month gt 'DEC95'
```

If you execute an ALLSTAT command, then table the MONTH dimension, you only see values of MONTH starting with January 1996. All variables dimensioned by MONTH are affected by this restriction, and maintenance of the MONTH dimension is not allowed, even if a PERMIT MAINTAIN command exists for MONTH.

Write permission is only meaningful for dimensions in providing write access to dimensioned objects. Consider, in addition to the read permission described above, the following write permission for MONTH.

```
describe month
```

```
DEFINE MONTH DIMENSION MONTH
LD Month
PERMIT READ WHEN month gt 'DEC95'
PERMIT WRITE WHEN month gt 'DEC96'
```

If the SALES variable has the following permission associated with it, then you can view sales figures for the BOSTON and CHICAGO districts in 1996 and 1997, but you can only modify those values for the BOSTON district in 1997.

```
describe sales
```

```
DEFINE SALES VARIABLE DECIMAL <MONTH PRODUCT DISTRICT>
LD Sales Revenue
PERMIT READ WHEN district eq 'BOSTON' or district eq 'CHICAGO'
PERMIT WRITE WHEN district eq 'BOSTON'
```

Designing a Database Security Application

The database permission programs PERMIT_READ and PERMIT_WRITE allow you to establish the permission conditions for a database and have them take effect when the database is attached. The permission associated with database objects is not in effect within the database permission programs. Your database permission programs must first execute explicit PERMIT or PERMITRESET commands if you want permission to be evaluated when you reference database objects within these programs.

Tip: When debugging a PERMIT_WRITE program, it is a good idea to temporarily rename it PERMIT_READ. This allows you to test your program logic within the context of a DATABASE ATTACH command. However, if the program fails, you can still attach the database with read/write access and edit your database permission program. Once you have tested it, you can rename it back to PERMIT_WRITE.

A typical approach to designing a database permission program would be first to check for the identity of the user attaching the database. The program could then determine if the user is the DBA, and if so, would grant the authority to modify the permission structure for the database. If the user is not the DBA, the program could then determine what authority level this particular user should have.

Consider that the following objects exist for security purposes in your database.

describe dba

```
DEFINE DBA VARIABLE BOOLEAN
LD If yes, permission can be modified
PERMIT WRITE WHEN DBA
PERMIT PERMIT WHEN DBA
```

describe userid

```
DEFINE USERID VARIABLE TEXT
LD Identification code of user of the database
PERMIT WRITE WHEN DBA
PERMIT PERMIT WHEN DBA
```

describe user

```
DEFINE USER DIMENSION TEXT
LD Valid Users of the Database
PERMIT MAINTAIN WHEN DBA
PERMIT PERMIT WHEN DBA
```

report user

```
BCD
RMS
OPL
WSM
```

describe authority

```
DEFINE AUTHORITY DIMENSION TEXT
LD Authority Classification for Users
PERMIT MAINTAIN WHEN DBA
PERMIT PERMIT WHEN DBA
```

report authority

```
CLASS1
CLASS2
```

describe authority_user

```
DEFINE AUTHORITY_USER RELATION AUTHORITY <USER>
LD Users with their Permission Classifications
PERMIT WRITE WHEN DBA
PERMIT PERMIT WHEN DBA
```

report authority_user

USER	AUTHORITY _USER
BCD	CLASS1
RMS	CLASS2
OPL	CLASS2
WSM	CLASS2

describe salespermission

```
DEFINE SALESPERMISSION VARIABLE BOOLEAN <DISTRICT AUTHORITY>
LD Accessible sales district values per authority classification
PERMIT WRITE WHEN DBA
PERMIT PERMIT WHEN DBA
```

```
report salespermission
```

```
-----SALESPERMISSION-----
-----DISTRICT-----
AUTHORITY      BOSTON  ATLANTA  CHICAGO  DALLAS  DENVER  SEATTLE
-----
CLASS1         YES    YES    YES    YES    YES    YES
CLASS2         YES    NO     NO     NO     NO     NO
```

Now you can use the security structures shown above to limit access to the SALES variable. Additional boolean variables similar to SALESPERMISSION could be defined for use in PERMIT commands for other variables.

```
dsc sales
```

```
DEFINE SALES VARIABLE DECIMAL <MONTH PRODUCT DISTRICT>
LD Sales Revenue
PERMIT READ WHEN SALESPERMISSION
PERMIT PERMIT WHEN DBA
```

Following is a description of a PERMIT_WRITE program for this database. The PERMIT_WRITE program determines who is using the database and what type of access authority that user has. This sample program illustrates a simplified approach to implementing a security structure for a database. You could implement more elaborate security checking, for example, by passing a password associated with a user ID to the following program and testing to see if its value is "XYZ."

```
DEFINE PERMIT_WRITE PROGRAM BOOLEAN
PERMIT READ WHEN DBA
PERMIT PERMIT WHEN DBA
PROGRAM
arg _userid text
dba = no
userid = _userid
if userid eq 'XYZ'
  then goto isdba
  else do
    if isvalue (user,userid)
      then do
        consider authority
        permit read when authority eq -
                                authority_user(user userid)
      doend
    else goto notvalid
  doend
return yes
isdba:
  dba = yes
  return yes
```

```
notvalid:
    show 'Not a valid identification code.'
    return no
END
```

Express Server runs this PERMIT_WRITE program before attaching the database. Express Server attaches the database with read/write access if and only if the PERMIT_WRITE program returns YES.

If the user is the DBA, then the permission structures within the database may be modified. When a user accesses the SALES variable, the permission established in the PERMIT_WRITE program goes into effect. The DBA and the Class 1 user can see and modify all SALES values. A Class 2 user can see and modify all SALES values for BOSTON only.

Chapter 10

Configuring for SQL and XCA Connections

This chapter describes how to set up for connecting to relational databases to perform dynamic SQL reach-through.

This chapter also describes how to use Administrator to configure a database that resides on an Express Server remote machine for use with Express Client products and Personal Express 5.x through XCA. It describes the tasks you perform to set up a Personal Express XCA connection to Express Server for you or your users.

This chapter describes the following:

- Setting up for SQL connections
- Setting up for a Personal Express XCA connection to Express Server
- Summary

Setting Up the System for SQL Connections

Chapter 5 describes how to set up the formulas you need to perform SQL reach-through, which allows data to be fetched from a relational database as client application users request it.

This section first describes how to set up an ODBC data source, to which later sections of this chapter refer, then how to prepare code for connecting to SQL databases.

Setting Up an ODBC Data Source

Before you can set up a system for SQL reach-through, you must set up an ODBC data source.

It is most likely for Administrator to be installed on a PC and Express Server to be installed on a remote machine. In this case, you must set up a 16-bit ODBC data source on your PC that has the same name and configuration settings as the 32-bit ODBC data source on the Express Server machine. To do so, use the following procedure.



To set up an ODBC data source:

1. Check with your system administrator to make sure Express Server is fully configured for SQL support.
2. Get information about the 32-bit ODBC data source that has been set up on the Express Server machine. (You will need to talk to the system administrator or the administrator of your data source.) The information you need is the name of the ODBC data source and its configuration settings.
3. On the PC where you have installed Administrator, set up a 16-bit ODBC data source that has the same name and the same configuration settings as the 32-bit ODBC data source on the server.

If Administrator is installed on Windows NT or Windows 95 and you have installed Personal Express 5.x on the same PC, you must also set up a 32-bit ODBC data source for Personal Express. Personal Express 5.x is a product available separately from Oracle. If you set up a 32-bit ODBC data source, Oracle recommends that you set up a system data source instead of a user data source. A system data source on an NT server is available to all users, regardless of the user ID used to establish the connection.



To set up a system data source, choose **System DSN** at the bottom of the Data Sources dialog box. This displays the System Data Sources dialog box, where you can add a system data source.

If Administrator is installed on the same machine as Express Server, you must set up a 16-bit ODBC data source for Administrator on the machine. Also, check with your system administrator to make sure Express Server is fully configured for SQL support and that a 32-bit ODBC data source has already been set up.

Preparing Code that Connects to the SQL Database

To allow users to request data to be fetched from a relational database, you must prepare code that connects to the SQL database, and you must include this code in the Pre-Attach Program and Post-Attach Program properties of the database that contains the SQL reach-through formulas. You can decide whether to display, in the client application, the SQL Login dialog box that prompts users for their IDs and passwords for connecting to SQL databases.

To understand what to include in a connection program, you must understand the temporary variables that allow the client application to communicate with the code in your connection programs. These variables are stored in XPDDCODE.DB and are described in Table 10.1. The table contains the names of dimension values, shown in parentheses, in the Function column for the XP_SQLACTION and XP_SQLSTATUS variables. These dimension values can be used in qualified data references (QDRs) to serve as constants.

To set the variables with dimension values used as constants, you use a QDR with two objects from XPDDCODE.DB. One object is a variable called XP_VAL. Another is a text dimension called XP_DEF. For example, suppose you want to set the XP_SQLACTION variable to tell your client application to connect to the SQL database. You can enter one of the following lines of code. The two lines accomplish the same task.

```
xp_sqlaction = 1
```

or

```
xp_sqlaction = xp_val(xp_def 'XP_ACTLOGIN')
```

Table 10.1 Temporary Variables for SQL Connections

Variable	Data Type	Function
XP_SQLACTION	INTEGER	Tells your connection code or the client application what action to take. You set the value of this variable. Possible values are: 0 = Do nothing. (XP_ACTNONE) 1 = Connect to the SQL database. (XP_ACTLOGIN) 2 = Give the user a message. (XP_ACTMSG) 3 = Detach the current database. (XP_ACTDETACH) 4 = Detach the database and give the user a message. (XP_ACTDETACHMSG)
XP_SQLLOGINMSG	TEXT	Holds the text of an informational message to display in the SQL Login dialog box. The text can contain up to 38 characters.
XP_SQLMESSAGE	TEXT	Holds the text of an informational message to display in a dialog box after the connection process.
XP_SQLPASSWORD	TEXT	Holds the user's password to access the relational database.
XP_SQLSTATUS	INTEGER	Tells your connection code or the client application the state of the connection process. The client application sets the value of this variable. Possible values are: 0 = Everything is proceeding normally. (XP_STSOK) 1 = There is already a connection. (XP_STSCONNECTED) 2 = The user canceled the SQL Login dialog box. (XP_STSCANCEL)
XP_SQLUSERNAME	TEXT	Holds the user ID to access the relational database.

Understanding the Sequence of Events

When client application users attach a database manually or when it is attached automatically and that database is meant to connect to an SQL source, the following sequence of events occurs:

1. The client application initializes the XP_SQLUSERNAME, XP_SQLPASSWORD, XP_SQLLOGINMSG, XP_SQLSTATUS, and XP_SQLACTION variables to NA or 0.
2. The client application runs the program specified in the database's Pre-Attach Program property. If you want the client application to display the SQL Login dialog box and if you want to display a message in that dialog box, enter the following commands in the pre-attach program.

```
xp_sqlaction = xp_val (xp_def 'XP_SQLLOGIN')
xp_sqlloginmsg = 'Connect to SQL database FRED'
```
3. The client application attaches the database. If it cannot attach the database, it skips the following steps.
4. If the user chooses **OK** in the SQL Login dialog box, the client application stores the user ID and password in the XP_SQLUSERNAME and XP_SQLPASSWORD variables. (The user cannot choose **OK** if the user ID field is blank). If the user chooses **Cancel**, the client application sets the XP_SQLSTATUS variable to 2.
5. The client application runs the program specified in the database's Post-Attach Program property. This program should contain code to accomplish the following:
 - Connect to the SQL database.
 - Handle nonzero values set in the XP_SQLSTATUS variable and set the XP_SQLACTION variable appropriately.
 - Set a value in the XP_SQLMESSAGE variable to display to the user (optional).
6. If the XP_SQLACTION variable is not set to 0, the client application takes the appropriate action, based on the variable's setting. Such actions might include sending a message to the user or detaching the Express database.

Using Two Administrator Programs for SQL Connections

The next section, “Sample Connection Programs,” provides examples of program code that you can use to connect to and disconnect from SQL databases. These examples call on two programs provided with Administrator — XP_SQLCNCT and XP_SQLDISCNCT. Before understanding the example programs, you must understand the XP_SQLCNCT and XP_SQLDISCNCT programs.

XP_SQLCNCT Program

As you begin the process of connecting to a SQL database, you use the XP_SQLCNCT program. This program returns a boolean value indicating whether the connection was successful. The program takes the following five arguments:

- **ODBC Data Source** — The name of the ODBC data source, which is described earlier in this chapter.
- **Relational dbms** — The type of the relational database to which you are connecting, such as Oracle.
- **NA** — This third argument is always NA.
- **User ID** — The user ID with which you make the SQL connection.
- **Password** — The password associated with the user ID.

The following example lines show uses for the XP_SQLCNCT program. The first example makes a connection to an Oracle database using a source called “Oracle 7016.” This database requires a user ID and password. The second example makes a connection to a Microsoft Access source that does not require a user ID or password.

```
xp_sqlcnct ('Oracle 7016', 'ORACLE', na, -  
           XP_SQLUSERNAME, XP_SQLPASSWORD)  
xp_sqlcnct ('MS Access')
```

XP_SQLDISCNCT Program

When you detach a database, you should disconnect from the SQL source. To do so, you use a program that calls the XP_SQLDISCNCT program. This program takes only one argument — the name of the ODBC data source, which is described earlier in this chapter.

The following example shows the XP_SQLDISCNCT program being used to disconnect from an Oracle database using a source called “Oracle 7016.”

```
xp_sqldiscnct ('Oracle 7016')
```

Tips for Using the Programs

When preparing programs that use XP_SQLCNCT and XP_SQLDISCNCT, you should keep the following tips in mind:

- When you distribute a query that requires a database using dynamic SQL reach-through, you must ensure that the end users have the appropriate ODBC drivers and access to the data source on the Express Server machine.
- Not all SQL sources require user ID and passwords. The pre-attach processor of the database will specify whether to display the SQL Login dialog box, and the temporary variables XP_SQLUSERNAME and XP_SQLPASSWORD will contain the values entered in the dialog box. You should pass these two variables as the User ID and Password arguments to the XP_SQLCNCT program.
- When running Administrator with Express Server, you must set up an ODBC source, because Administrator makes an ODBC connection to create the dynamic SQL reach-through formulas to import SQL data. This ODBC source must be identical to the ODBC source on the server.

Sample Connection Programs

Figure 10.1 provides a sample login program for starting the connection between a client application session and a SQL database. This program displays the SQL Login dialog box. This program name would be set as a database's Pre-Attach Program property value.

```
DEFINE SQL.PRE PROGRAM
PROGRAM
" Set the SQL Action to prompt with the SQL Login dialog box
XP_SQLACTION = XP_VAL (XP_DEF 'XP_ACTLOGIN')
XP_SQLLOGINMSG = 'Enter Oracle Userid/password'
END
```

Figure 10.1 Sample Pre-Attach program for a SQL connection.

Figure 10.2 provides a sample program for continuing to establish the connection between a client application session and a SQL database that began with the Pre-Attach program. This program name would be set as a database's Post-Attach Program property value.

```

DEFINE SQL.POST PROGRAM
PROGRAM
trap on ERROR noprint
pushlevel 'SQL.POST'
push SQLMESSAGES
SQLMESSAGES = YES

" Status is OK, connect to the SQL source
if XP_SQLSTATUS eq XP_VAL (XP_DEF 'XP_STSOK')
then do
  " Connect to the SQL source
  if XP_SQLCNCT ('Oracle 7016', 'ORACLE', na, -
    XP_SQLUSERNAME, XP_SQLPASSWORD)
  then do
    " Connection established, display message for the user
    XP_SQLACTION = XP_VAL (XP_DEF 'XP_ACTMSG')
    XP_SQLMESSAGE = 'Oracle SQL Connection established.'
  doend
  else do
    " Connection failed, detach the database and tell the user
    XP_SQLACTION = XP_VAL (XP_DEF 'XP_ACTDETACHMSG')
    XP_SQLMESSAGE = joinchars ('Connection Failed: ' -
      sqlerrm)
  doend
doend

" Status is connected, display message for the user
else if XP_SQLSTATUS eq XP_VAL (XP_DEF 'XP_STSCONNECTED')
then do
  XP_SQLACTION = XP_VAL (XP_DEF 'XP_ACTMSG')
  XP_SQLMESSAGE = 'The SQL connection is already made.'
doend

" Status is cancel, detach the database and tell the user
else if XP_SQLSTATUS eq XP_VAL (XP_DEF 'XP_STSCANCEL')
then do
  XP_SQLACTION = XP_VAL (XP_DEF 'XP_ACTDETACHMSG')
  XP_SQLMESSAGE = 'User canceled from the login dialog.'
doend

DONE:
poplevel 'SQL.POST'
return
ERROR:
XP_SQLACTION = XP_VAL (XP_DEF 'XP_ACTDETACHMSG')
XP_SQLMESSAGE = joinlines('Postprocessor Error:' errortext)
goto DONE
END

```

Figure 10.2 Sample Post-Attach program for a SQL connection.

Figure 10.3 provides a sample program for breaking the connection between a client application session and a SQL database. This program

name would be set as a database's Pre-Detach or Post-Detach Program property value.

```
DEFINE SQL.DETACH PROGRAM
PROGRAM
" Disconnect form the SQL Source
call XP_SQLDISCNCT ('Oracle 7016')
END
```

Figure 10.3 Sample detach program for a SQL connection.

Setting Up a Personal Express XCA Connection to Express Server

If you have Personal Express 5.x, you may set up a Personal Express XCA connection to the remote machine on which Express Server runs. Personal Express 5.x is provided as a separate product; it is not part of Express Server 6.0 or later.

You are likely to need the information in this section if you have used Personal Express 5.x with earlier versions of Express Server. With Personal Express 5.x, you can create a *local database*, which is a database that does not reside on a remote server machine. You can give copies of local databases (and their associated application databases, such as XPDDDATA.DB) to client application users. You or your users can attach a local database, then set up a Personal Express XCA connection to the Express Server machine. Once connected to the Express Server machine, you or your client application users can attach a database that resides on the remote machine and use it with a local database.

The following sections describe how to:

- Prepare for connecting to a remote system
- Set the MDBSource property
- Manage security
- Understand how the Express Client Selector works with remote data from XCA

Preparing to Make a Personal Express XCA Connection to a Remote System

To use Personal Express 5.x with remote data, you must provide the following:

- A communications link supported by Personal Express 5.x and the Express Communications Architecture (XCA).
See the *Personal Express 5.0 Communications Guide* for information about communications.

- One or more login programs for the remote system to set up the connection, log in, make sure Express Server is running on the remote machine, attach the database, and open an XCA session. The programs can take any form and use whatever error recovery mechanisms are appropriate. A simple program might just disconnect at an error. A more complex program might return to a previous stage of the login process and try again. The names of the programs are stored as the pre-attach and post-attach programs for the configured local database.

You can decide whether a user connects automatically to the remote system or whether Express Client products display the Login dialog box where the user can specify a user ID and password.

- A logout program that exits remote Express, logs off the remote system, and disconnects. You can specify the name of this program in the local database's Pre-Detach Program property.

To understand what to include in a login program, you must understand the temporary variables that allow Express Client products to communicate with the code in your login programs. These variables are stored in XPDDCODE.DB and are described in Table 10.2. The table contains the names of dimension values, shown in parentheses, in the Function column for the XP_XCAACTION and XP_XCASTATUS variables. These dimension values can be used in qualified data references (QDRs) to serve as constants, as described in the next section “Using Constants.”

Using Constants

When you are writing login programs, you set the values of the temporary variables described in Table 10.2. To set or read the values for the XP_XCAACTION and XP_XCASTATUS variables, you can use the integers shown in the table. Or you can use the dimension values shown in parentheses in the table as constants. When you read program code, constants often provide a more meaningful setting for the variables than integers.

To set the variables with dimension values used as constants, you use a QDR with two objects from XPDDCODE.DB. One object is a variable called XP_VAL. Another is a text dimension called XP_DEF. For example, suppose you want to set the XP_XCAACTION variable to tell your login code or Express Client product to log in to the remote system. You can enter one of the following lines of code. The two lines accomplish the same task.

```
xp_xcaaction = 1
```

or

```
xp_xcaaction = xp_val (xp_def 'XP_ACTLOGIN')
```

Table 10.2 Temporary Variables for Remote Login

Variable	Data Type	Function
XP_XCAACTION	INTEGER	Tells your login code or Express Client products what action to take. You set the value of this variable. Possible values are: 0 = Do nothing. (XP_ACTNONE) 1 = Log in to the remote system. (XP_ACTLOGIN) 2 = Give the user a message. (XP_ACTMSG) 3 = Detach the current database. (XP_ACTDETACH) 4 = Detach the database and give the user a message. (XP_ACTDETACHMSG) 5 = Prompt the user to add a domain. (XP_ACTLOGINDOMAIN)
XP_XCADOMAIN	TEXT	Holds the name of the domain to which your user ID belongs. This value is relevant only if you are connecting to Express 6.0 or later, not Express 5.x. The value is required only when you do not have a user ID on the specific computer on which Express is running, but you do have an ID in the domain to which the computer belongs.
XP_XCALOGINMSG	TEXT	Holds the text of an informational message to display for the user in the Login dialog box. The text can contain up to 38 characters.
XP_XCAMESSAGE	TEXT	Holds the text of an informational message to display for the user in a dialog box after the login process.
XP_XCAPASSWORD	TEXT	Holds the password on the remote system.
XP_XCASTATUS	INTEGER	Tells your login code or Express Client products the state of the login process. Express Client products set the value of this variable. Possible values are: 0 = Everything is proceeding normally. (XP_STSOK) 1 = There is already a connection; the XOPEN function is true. (XP_STSCONNECTED) 2 = The user canceled from the Login dialog box. (XP_STSCANCEL)
XP_XCAUSERNAME	TEXT	Holds the user ID of the remote system.

Understanding the Sequence of Events

When Express Client users attach a local database manually or when it is attached automatically, the following sequence of events occurs:

1. The Express Client product initializes the XP_XCAUSERNAME, XP_XCAPASSWORD, XP_XCALOGINMSG, XP_XCASTATUS and XP_XCAACTION variables to NA or 0.
2. The Express Client product runs the program specified in the database's Pre-Attach Program property. If you want the Express Client product to display the Login dialog box and if you want to display a message in that dialog box, enter the following commands in the pre-attach program.

```
xp_xcaaction = xp_val (xp_def 'XP_ACTLOGIN')  
xp_xcaloginmsg = 'Login to server FRED'
```

3. The Express Client product attaches the local database. If it cannot attach the database, it skips the following steps.
4. If the XP_XCAACTION variable is set to 1 or 5 and the XOPEN function is false (indicating that there is no XCA connection), the Express Client product displays the Login dialog box. If the XOPEN function is true, the Express Client product sets the XP_XCASTATUS variable to 1, indicating that there is already a connection.
5. If the user chooses **OK** in the Login dialog box, the Express Client product stores the user ID in the XP_XCAUSERNAME variable, the password in the XP_XCAPASSWORD variable, and the domain name in the XP_XCADOMAIN variable. (The user cannot choose **OK** if the user ID is blank). If the user chooses **Cancel**, the Express Client product sets the XP_XCASTATUS variable to 2.

Note that the domain name is relevant only if you are connecting to Express 6.0 or later, not Express 5.x. The domain name is required only when you do not have a user ID on the specific computer on which Express is running, but you do have an ID in the domain to which the computer belongs.

6. The Express Client product runs the program specified in the local database's Post-Attach Program property. This program should contain code to accomplish the following:
 - Log in to the remote system and attach a database.

- Handle nonzero values set in the XP_XCASTATUS variable and set the XP_XCAACTION variable appropriately.
 - Set a value in the XP_XCAMESSAGE variable to display a message to the user (optional).
7. If the XP_XCAACTION variable is not set to 0, the Express Client product takes the appropriate action, based on the variable's setting.

Sample Connection Programs

Figure 10.4 provides a sample login program for connecting via XCA from an Express Client session to a remote system running Express. This program name would be set as a database's Pre-Attach Program property value.

```
DEFINE XCA.PRE PROGRAM
PROGRAM
" Set the XCA Action to prompt with the login dialog box
XP_XCAACTION = XP_VAL (XP_DEF 'XP_ACTLOGINDOMAIN')
XP_XCALOGINMSG = 'Enter Remote Password'
END
```

Figure 10.4 Sample Pre-Attach program for an XCA connection.

Figure 10.5 provides a sample program for continuing to establish the XCA connection that began with the Pre-Attach program. This program name would be set as a database's Post-Attach Program property value.

Figure 10.6 provides a sample program for breaking the XCA connection. This program name would be set as a database's Pre-Detach or Post-Detach Program property value.

```
DEFINE XCA.POST PROGRAM
PROGRAM
trap on ERROR noprint
pushlevel 'XCA.POST'
" Status of OK, connect to the XCA source
if XP_XCASTATUS eq XP_VAL (XP_DEF 'XP_STSOK')
then do
  comset type socket
  connect -
    server 'server1' -
    id XP_XCAUSERNAME -
    password XP_XCAPASSWORD -
    portno '6548' -
    encrypt 'yes' -
    domain XP_XCADOMAIN
  xopen
  execute 'dtb attach remote.db'
doend
" Status is connected, give message to the user
else if XP_XCASTATUS eq XP_VAL (XP_DEF 'XP_STSCONNECTED')
then do
  XP_XCAACTION = XP_VAL (XP_DEF 'XP_ACTMSG')
  XP_XCAMESSAGE = 'The XCA connection is already made.'
doend
" Status is cancel, detach the database and tell user
else if XP_XCASTATUS eq XP_VAL (XP_DEF 'XP_STSCANCEL')
then do
  XP_XCAACTION = XP_VAL (XP_DEF 'XP_ACTDETACHMSG')
  XP_XCAMESSAGE = 'The XCA connection failed because the -
    user canceled from the login dialog.'
doend

DONE:
poplevel 'XCA.POST'
return

ERROR:
XP_XCAACTION = XP_VAL (XP_DEF 'XP_ACTDETACHMSG')
XP_XCAMESSAGE = joinlines('Postprocessor Error:' errortext)
goto DONE
END
```

Figure 10.5 Sample Post-Attach program for an XCA connection.

```
DEFINE XCA.DETACH PROGRAM
PROGRAM
if xopen
    then xclose
disconnect
END
```

Figure 10.6 Sample detach program for an XCA connection.

Setting the MDBSource Property

The MDBSource property holds the name of the remote counterpart of an object in a Personal Express 5.x database. The object might be a data variable (e.g., SALES), a label variable (e.g., NAME.PRODUCT), a relation (e.g., JOBCODE.EMP), or a dimension (e.g., PRODUCT). For example, the SALES variable in Personal Express 5.x might have its MDBSource property set to “REMOTE.SALES.” The remote object need not have the same name as the Personal Express 5.x object. The setting of this property is especially important for use with the Selector in Express Client products, as described later in this chapter.

You can specify a value for the MDBSource property in Administrator using the Inspector. For information on setting database properties, see Chapter 5.

Managing Security

The Express Client product can be set such that users can access the Personal Express 5.x window. Therefore Administrator restricts access to the Express variables that store the user ID and password. In particular, Administrator does the following:

- Sets the XCA_PASSWORD, XCA_USERNAME, and XCA_DOMAIN variables to NA immediately after Step 6 described in the section “Preparing to Connect to a Remote System” earlier in this chapter.
- Sets the Express option IFCOPY to FALSE immediately before storing the user ID and password from the Login dialog box in the Express

variables. This prevents anyone from viewing the SET commands in the Personal Express 5.x window.

For increased security, you should strip the code of your login programs so Express Client users cannot examine or change the programs. However, be sure to keep the unstripped programs in a database you do not release to users.

The Selector and Remote Data

When Express Client users work with the Selector, they perform queries on Express data. When users are connected to a remote database through XCA, it is more efficient for Express to execute queries on the remote system when appropriate, rather than moving large amounts of data to Personal Express 5.x and executing queries there. The following Selector tools execute queries on the remote system when necessary:

- Match
- Attribute
- Exception Tool (“Numeric” option) and (“Measure” option)
- Top/Bottom
- Sort

The following sections describe how the Selector uses the MDBSource property and how the Selector tools use data from a remote system.

Using the MDBSource Property

When the Selector prepares to query data for an object, it checks the setting of that object’s MDBSource property to determine if it has a remote source. If so, the query is normally executed on the remote system, rather than in Personal Express 5.x. The resulting status is transferred to Personal Express 5.x, but not the object’s values.

For example, suppose a variable called SALES is dimensioned by MONTH, PRODUCT, and DISTRICT and has its MDBSource property set to REMOTE.SALES. The user executes the Top/Bottom query “Find the top 10 products based on sales for month M and district D.” The Top/Bottom Tool first limits MONTH and DISTRICT on the remote

system to the values M and D and then executes the following command on the remote system.

```
limit product to top 10 basedon remote.sales
```

For this example to work properly, the following conditions must be met:

- REMOTE.SALES and its dimensions must be defined similarly to their Personal Express 5.x counterparts. Their names can differ, but their data types and dimensionality must be the same. However, SALES can be a formula and REMOTE.SALES can be a variable.
- SALES must have its MDBSource property set to the name of its remote counterpart. If the MDBSource property is not set for any of its dimensions, they are assumed to have remote counterparts with the same names.
- All values of the Personal Express 5.x dimensions MONTH, PRODUCT, and DISTRICT must exist in their remote counterparts. The remote dimensions, however, can have additional values.

Match Tool

The Match tool searches for a certain string in a dimension's value or label variable, depending on a Selector option. If the label variable is cached and the MDBSource property is set for its cache formula, the match query is executed on the remote system.

Attribute Tool

An attribute can be either a relation or a conjoint dimension linking two dimensions. Normally, the attribute would not be cached in Personal Express 5.x; however, it still must be *defined* in Personal Express. For example, suppose there are PRODUCT and COLOR dimensions on the remote system and an attribute relation COLOR.PRODUCT, which specifies the color of each product. The *values* of COLOR.PRODUCT need not exist in Personal Express 5.x; however, both COLOR and COLOR.PRODUCT must be defined in Personal Express and must have their MDBSource properties set. COLOR must also be populated so that the user can select the values of this attribute. The actual query (e.g., `limit`

product to color . product) is executed on the remote system, and no values of COLOR.PRODUCT are downloaded to Personal Express 5.x.

Exception Tool (“Numeric” Option), Top/Bottom Tool, and Sort Tool

These tools limit a dimension based on values of a single measure, for specific values of the measure’s other dimensions. The general mechanism is described in the SALES example above. There will typically be both a cache formula and a conjoined cache variable for the measure. The cache formula is the object that the user selects and that should have its MDBSource property set appropriately.

Exception Tool (“Measure” Option)

This tool limits a dimension to values for which two different measures satisfy a certain condition (e.g., one is more than a certain amount greater than the other). The query is executed on the remote system only if both measures have remote counterparts. If only one measure has its MDBSource property set, the query is executed in Personal Express 5.x, and the data is downloaded from the remote system.

Appendix A

Administrator Installation

This appendix describes how to install Administrator. It contains the following sections:

- Hardware and software requirements
- Installing Administrator
- Starting and exiting Administrator

Hardware and Software Requirements

To install and use Administrator, your PC must meet the following hardware and software requirements:

- DOS Version 5.0 or later.
- Microsoft Windows version 3.1, Microsoft Windows NT version 3.51, or Microsoft Windows 95.
- Access to Express Server Version 6.0 or later on an NT workstation. You cannot run the Excel Add-In or other Express Client products with Express Server unless you follow the special connection instructions in Chapter 2.
- If you intend to import data from a relational database that complies with Microsoft's Open Database Connectivity (ODBC), you must have the ODBC support and appropriate ODBC drivers installed (32-bit *and* 16-bit).
- At least 4MB of disk space for Administrator files if you choose no options at installation. At least 6MB of disk if you include options.

- If you plan to install Administrator on a LAN server for shared use, you must have write access to that server.
- If you are running a LAN-based version of Windows, you must have write access to the Windows system directory.
- To use Administrator with Express Server, you must also install the following:
 - Remote SNAPI. For information about installing remote SNAPI, as well as its configuration and administration, see the *Oracle Express Server API Guide*.
 - RPC runtime support.
 - TCP/IP support on the client machine.

Installing Administrator

This section describes the steps you take to install and begin to use Administrator. Some steps are required, while some are optional.

Important: Administrator can be installed only for a single user, whether it is installed on a PC or a LAN.



To install Administrator:

1. Your system administrator should install Express Server on a remote machine. This step must precede the other steps. To use the Excel Add-In or other Express Client products with Express Server, be sure to follow the special connection instructions in Chapter 2.

Important: This appendix describes the installation process for Administrator. You must also follow additional connection steps to connect to Express Server. See Chapter 2 for detailed instructions about configuring Administrator to connect to Express Server 6.0.

2. Optional. Install ODBC drivers if you plan to import data from a relational database that complies with ODBC.
3. Optional. Install the Acrobat reader, as described in the section “Reading Documents Online” later in this appendix.

Note: Although this step is optional, Oracle Corporation strongly recommends that you install the reader so you can read and print important documents, such as ADDINS.PDF (instructions for using product add-ins).

4. Install Administrator. See the following sections for detailed instructions for installing from the CD or installing from a shared LAN area.
5. Optional. Use your favorite text editor to read RELNOTES.TXT to learn about new features and known bugs.
6. Start Administrator and configure existing databases or create new databases. Technically, configuring is optional. However, if you plan to use Express Client products with Express Server, some of the most powerful features of these client application products will not be available unless you configure the databases. For information about creating and configuring databases, see Chapters 4 and 8.

Installing from the CD

To install Administrator, run the Setup program found in the appropriate DISK1 subdirectory on the CD, as described in the following procedure.



To install from the CD:

1. Start Microsoft Windows and insert the CD into the appropriate drive.
2. Run the file:
`x:\xadmin\disk1\setup.exe`
3. Choose **OK** to continue.
4. Follow the Setup program's instructions. If you are unsure how to respond to a dialog box, choose **Help** for additional information. The Setup program installs the product and places its icons in the Express Client program group. If the program group does not exist, the Setup program creates it.

Copying Installation Files

If you want to install additional licensed copies of Administrator from the single CD, you can copy the installation files onto disks or onto a shared LAN area to make distribution easier. Users can run the Setup program from the LAN or from the CD to install Administrator onto their PCs, as described in the following section “Installing from a Shared LAN Area or Disks.” This section describes how to copy the files from the CD to a shared LAN area or to disks.



To create disks from the CD:

1. Use a file manager utility to select the directory on the CD that contains the Administrator files you wish to copy to disks. You will need a blank disk for each DISKx subdirectory.
2. Insert a blank disk into your disk drive.
3. Select all the files under each DISKx subdirectory on the CD and copy them to the blank disk. Do *not* copy the entire directory, just the files. Place the files from each subdirectory onto a different disk, until all the DISKx subdirectories have been copied. You will have 4 Administrator disks.
4. Label all disks with the correct product name and disk number.

To install from the disks, follow the instructions in the earlier section “Installing from a Shared LAN Area or Disks.”



To copy files to the LAN so users can install from there:

1. Use a file manager utility to select the directory on the CD that contains the Administrator files.
2. Be sure there is enough space on the LAN to copy all the files for Administrator.
3. Copy the entire product directory from the CD to the LAN area, including all of the DISKx subdirectories.

To install from the LAN, follow the instructions in the next section “Installing from a Shared LAN Area or Disks.”

Installing from a Shared LAN Area or Disks

To facilitate distribution to multiple licensed users, you can copy the files on the CD to a shared LAN area or to disks and allow users to install from there, as described in the following procedures. For information about copying the files, see the section “Copying Installation Files” earlier in this appendix.



To install from a shared LAN area:

1. Run the Setup program for Administrator. For example, if the files were copied to DISKx subdirectories of I:\ORACLE\EXPRESS\XADMIN, you would type the following:

```
i:\oracle\express\xadmin\disk1\setup.exe
```

2. Choose **OK** to continue.
3. Follow the Setup program's instructions. If you are unsure how to respond to a dialog box, choose **Help** for additional information. The Setup program installs Administrator and places its icons in the Express Client program group. If the program group does not exist, the Setup program creates it.



To install from disks:

1. Start Microsoft Windows and insert the first disk for Administrator into a disk drive on your PC.
2. Run the Setup program. In the Command Line box, type
a:\setup.exe (or b:\setup.exe, if you are using the b drive).
3. Choose **OK** to continue.
4. Follow the Setup program's instructions. If you are unsure how to respond to a dialog box, choose **Help** for additional information. The Setup program installs Administrator and places its icons in the Express Client program group. If the program group does not exist, the Setup program creates it.

Reading Documents Online

The manuals for the Express Server products and Administrator are available on the installation CD and can be read online. To read these .PDF documents online, you use the Acrobat reader, which you must install from the CD.



To install the Acrobat reader:

1. Insert the CD into the appropriate drive.
2. Run the Acroread program. Type `x:\reader\acroread.exe`, where `x` identifies your CD-ROM drive.
3. Follow the instructions on your screen. Installation requires approximately 2MB of free space in your \TEMP directory. If the required space is not available, installation will terminate with a *0003* error.



To read a document online:

1. Insert the CD into the appropriate drive.
2. Start the Acrobat reader by double-clicking on its icon.
3. From the File menu, choose **Open**.
4. Specify the full path name of CONTENTS.PDF. This file contains a table of contents that includes jumps, which you can use to navigate to any of the available documents.

or

Specify the full path name of one of the .PDF document files. All files use the Acrobat file extension, .PDF.

Once the file is open in the reader, you can use Acrobat's facilities to jump to a particular chapter or to search for the information you want. For more information about how to use the Acrobat reader, choose **Acrobat Reader Help** from the Help menu.



For information about how to navigate among multiple Help files, click on **Using Multiple Help Systems** on the main contents screen of the Administrator Help system.

Starting and Exiting Administrator

▶▶ To start Administrator, double-click on its icon in the Express Client program group.

You must edit the settings in the XADMIN.INI file before you can connect to Express Server. See Chapter 2 for details.

▶▶ To exit Administrator, from the File menu, choose **Exit**.

Appendix B

Using Conjoint Dimensions

Chapter 9 describes sparse data and how to define sparse variables with Administrator as a way of making your database more efficient. This is generally the better method of handling sparse data.

However, you can also use conjoint dimensions to handle sparse data.

A *conjoint dimension* is a special type of dimension whose values correspond to specific combinations of values from one or more other dimensions called *base dimensions*. Normally, the conjoint dimension would only have those combinations of base dimension values for which you have data.

This appendix describes how to:

- Define and maintain conjoint dimensions
- Convert a conjoint dimension to a composite
- Convert a composite to a conjoint dimension

Defining and Maintaining Conjoint Dimensions

One way to eliminate NA values and make data variables smaller is to use conjoint dimensions instead of composites.

Composites give you a quick and easy way to handle sparse variables. Conjoint dimensions give you finer control in working with sparse

variables, but you will then have to take extra steps to maintain your conjoint dimensions.

For example, suppose you want to limit the dimensions of a sparse variable. If that variable is dimensioned by a composite, you limit the base dimensions of that composite (such as limiting the PRODUCT dimension to 'SPORTSWEAR'). However, if the variable is dimensioned by a conjoint dimension, you can limit the actual combinations of the conjoint dimension (such as <'SPORTSWEAR' 'BOSTON'>). If you find you need this degree of control, you might consider using conjoint dimensions instead of composites. Be aware, though, that you will be sacrificing the ease of use that composites provide.

When you define a conjoint dimension, Express automatically defines relations between the conjoint dimension and the base dimensions from which it is derived. Therefore, you can select and retrieve data based on the base dimension values.

For example, suppose you have a variable SALES, which is dimensioned by MONTH, PRODUCT, and DISTRICT, as follows.

```
define sales decimal <month product district>
```

Each district only sells certain products and never has data for other products. The following report shows the values in the SALES variable.

-----SALES-----						
	JAN95	FEB95	MAR95	APR95	MAY95	JUN95

BOSTON:						
COFFEE	31,462.80	58,343.75	54,158.84	45,459.55	63,727.98	52,232.05
SUGAR	45,633.38	66,960.37	51,201.41	41,119.10	66,985.30	34,103.41
GRITS	NA	NA	NA	NA	NA	NA
GRANOLA	38,353.49	44,890.01	32,705.20	59,889.76	26,695.59	25,807.17
ATLANTA:						
COFFEE	62,373.42	37,698.17	39,944.58	42,854.71	31,815.47	26,659.09
SUGAR	NA	NA	NA	NA	NA	NA
GRITS	55,075.33	64,352.74	36,571.66	31,048.07	28,430.80	69,612.45
GRANOLA	NA	NA	NA	NA	NA	NA

CHICAGO:							
COFFEE	28,302.61	31,301.34	53,122.95	32,510.59	50,884.83	33,123.26	
SUGAR	43,460.88	22,743.96	45,991.54	39,841.34	25,657.83	50,375.86	
GRITS	NA	NA	NA	NA	NA	NA	NA
GRANOLA	NA	NA	NA	NA	NA	NA	NA

To eliminate the NA values and make the data variable smaller, you can define a conjoint dimension whose values are combinations of the values of the PRODUCT and DISTRICT dimensions.

```
define proddist dimension <product district>
```

Then you maintain the new dimension by adding only those combinations of product and district values that have data. If you use the MAINTAIN command, enclose each combination in angle brackets. In this example, there are only seven combinations that have data.

```
maintain proddist add <'COFFEE' 'BOSTON'>  
<'COFFEE' 'ATLANTA'> <'COFFEE' 'CHICAGO'>  
<'SUGAR' 'BOSTON'> <'SUGAR' 'CHICAGO'>  
<'GRITS' 'ATLANTA'> <'GRANOLA' 'BOSTON'>
```

Now you can define a variable dimensioned by PRODDIST for the sales data. Because a conjoint dimension is automatically related to its component dimensions, a single assignment statement transfers the non-NA data to the new variable.

```
define sales2 variable decimal <month proddist>  
sales2 = sales  
delete sales  
rename sales2 sales
```

After you assign the values from the original variable to the new one, you can delete the original variable. You can rename the new variable to the name of the original variable if you wish to keep the old name. In the example above, SALES was converted to SALES2, which has more than 40 percent fewer cells. The original variable, SALES, was deleted, and the new variable, SALES2, was renamed SALES.

You need to build the combinations to be added to the conjoint dimension. You can use the FILEREAD command to read data from the input file and add values to the base dimensions and the conjoint dimension.

If you have an input file with product names in column one and district names in column nine, you can add new values to the PRODUCT, DISTRICT, and PRODDIST dimensions with the following FILEREAD command. New values are added first to the base dimensions, then the combination is added to the conjoint dimension.

```
fileread unit append proddist = <w 8 append product, w 8 append district>
```

Converting Conjoint Dimensions to Composites

If you have used a previous version of Express, you may have already created conjoint dimensions. You may want to convert them to composites to take advantage of the ease of use that composites provide.

You can convert a conjoint dimension to a named composite. However, there are several prerequisites:

- If an Express object returns a conjoint dimension value, you cannot convert that conjoint dimension to a composite. For example, if your database contains a program whose return value is a conjoint dimension value, you cannot convert that conjoint dimension to a composite.
- You cannot convert a NOHASH conjoint dimension. You can, however, use the CHGDFN command to change the conjoint dimension to BTREE or HASH, and then you can convert it to a composite.
- If the definition of the conjoint dimension contains both a dimension and a second conjoint dimension that contains the same dimension in its definition, you cannot convert the first conjoint dimension. For example, suppose the definition of a conjoint dimension named MYDIM contains a dimension named MONTH and a conjoint dimension named YOURDIM. The definition of the conjoint dimension YOURDIM contains the same MONTH dimension. Therefore, MONTH appears twice in the definition of MYDIM, so MYDIM cannot be converted to a composite.
- If there are permissions set on a conjoint dimension, you cannot convert it to a composite. You can remove the permissions, and then convert the

conjoint dimension to a composite. You can then set permissions on the base dimensions of the composite, if you wish.

- The conjoint dimension must have been defined with the DEFINE command, not with Administrator.



To convert a conjoint dimension to a named composite, use the COMPOSITE keyword with the CHGDFN command. Use the following syntax.

```
chgdfn conjoint-name COMPOSITE
```

Conjoint-name is the name of the existing conjoint dimension. The COMPOSITE keyword specifies that you are converting a conjoint dimension to a composite.

When you convert a conjoint dimension to a composite, the conjoint name becomes the name of the composite. In other words, when a conjoint is converted to a composite, it keeps its name.

See the entry for CHGDFN in the Express Language Help system for more information about using CHGDFN.

Converting Composites to Conjoint Dimensions

If you are willing to take the extra steps to maintain conjoint dimensions in exchange for finer control, you can convert composites to conjoint dimensions.

You can convert either a named composite or an unnamed composite to a conjoint dimension. There are no prerequisites; you can convert any existing composite to a conjoint dimension.



To convert a named composite to a conjoint dimension, use the DIMENSION keyword with the CHGDFN command. Use the following syntax.

```
chgdfn composite-name DIMENSION
```

Composite-name is the name of the existing composite. The DIMENSION keyword specifies that you are converting a named composite to a conjoint dimension.

When you convert a composite to a conjoint dimension, it keeps its name. The composite name becomes the conjoint dimension name.



To convert an unnamed composite to a conjoint dimension:

1. Use the RENAME command to change the unnamed composite into a named composite. For example, suppose the unnamed composite exists in a sparse variable definition as “sparse<product district>.” To change this unnamed composite to a composite named PRODDIST, use the following command.

```
rename sparse<product district> proddist
```

2. Use the CHGDFN command to change the named composite, PRODDIST, into a conjoint dimension.

```
chgdfn proddist DIMENSION
```

Index

Dimension Values property, 87
= command, 177

A

Administrator
 defined, 32
 INI file, 21
 installing, 227, 228
 running on same machine as
 Express, 28
 starting, 36
advantages of configuring, 34
aggregations in hierarchies, 134
ALLSTAT command, 153
application database files, 20, 35
application databases
 locating, 84
ASCII data
 importing, 104
Attach Order property, 78
attaching databases, 76
Attribute tool, 225
authentication
 domain, 25
 host, 26
AutoAttach property, 78

B

base dimensions, 237
browser, *See* Database Browser

C

cell
 defined, 156
CHGDFN command, 181
 using the COMPOSITE keyword,
 241
 using the DIMENSION keyword,
 242
 using the SEGWIDTH keyword,
 181
client applications, 17
 .INI files, 21
 preparing databases for, 34, 124
 running programs at shutdown, 142
commands
 executing, 149
compilation errors
 fixing, 115
Compiled property, 94
compiling
 programs, 114

composites

- aggregating hierarchical data, 187
- assigning names to unnamed, 174
- changing named to unnamed, 175
- converting to conjoint dimensions, 241
- defined, 12, 165
- defining a variable with multiple, 167
- defining single-dimension, 169
- how to delete base dimension values, 184
- how variables can share, 165
- limiting base dimensions, 185
- named, 172
- named, copying the definition, 175
- named, creating, 173
- specifying segment size, 181
- unnamed, 171

configuring databases

- check list, 124
- default selections, 139
- for remote data, 207
- hiding dimensions, 126
- installing databases, 77
- installing dimensions, 86
- installing measures, 92
- saved selections, 138
- shutdown programs, 142
- uninstalling dimensions, 128
- uninstalling measures, 128
- using hierarchies, 128–38

conjoint dimensions, 237

- converting to a named composite, 240
- defined, 237
- defining, 239
- maintaining, 239

connecting

- to Express Server, 19

connections

- for remote data, 217

connections to server

- security options, 25
- specifying one or more, 23
- specifying the Express Server version, 23
- supporting multiple, 24

constants, 218

controlled sparsity, 159

- defined, 158

COPYDFN command, 175

copying database objects, 117

creating

- dimensions, 85
- hierarchies, 134
- saved selections, 138
- valuesets, 112

current system language, 79–82

D

data, 2

- accessing performance, 181
- adding to a variable, 67
- assigning to sparse variables, 177
- exporting to EIF files, 108
- how stored, 155
- importing, 103
- importing ASCII, 104
- importing EIF, 106–8
- importing from a relational database, 106
- importing from an Express database, 107
- loading performance, 181

- multidimensional, 4
- remote, 207
- restricting access, 196
- rotating, 90
- sparse, 158
- viewing selected, 150
- data databases
 - defined, 36
 - locating, 83
- Data Dictionary
 - defined, 34
 - structures for hierarchies, 131, 134
- data loader
 - creating a SQL, 106
 - creating an ASCII, 105
- Data Reader
 - maintaining conjoint dimensions, 239
- data slice, 4
- data storage, 7
- Data Type property, 87
- data variables, 13
- database
 - defined, 2
 - defining a new, 51
 - integrity, 8
 - password, 195
 - upgrading an Express 5.x, 70
- Database Browser, 40
 - choosing display, 121
 - displaying search results, 121
- DATABASE command
 - attaching a database with
 - permission, 195, 196, 203
- database dictionary, 49
- database files
 - needed by Administrator, 35
 - saving space with formulas, 157
- DATABASE function, 175
- database objects
 - copying, 117
 - deleting, 118
 - exporting definitions, 108
 - how to define, 51
 - how to view all, 50
 - importing definitions, 106–8
 - renaming, 117
 - searching for, 121
 - setting properties, 119
- database permission programs, 195, 196, 203
- Database property, 87, 94
- databases
 - advantages of configured, 34
 - attaching, 76
 - attaching automatically, 83
 - attaching with permission, 195, 196, 203
 - backup, 190
 - configuring for client applications, 124
 - creating security applications for, 195, 196, 203
 - demo, 36
 - detaching, 82
 - efficient design, 154
 - installing, 77
 - maintaining, 75
 - making smaller, 188
 - modifying, 75
 - multifile, 192
 - preparing for use in clients, 34
 - properties, 78
 - rebuilding, 188
 - remote, 207
 - saving disk space, 157
 - saving one in multiple files, 192
 - setting properties, 120

- stored in more than one file, 192
- uninstalling, 127
- units of storage, 155
- data-driven formulas, 93
- DataType property, 94
- debugging programs, 116
- default number formats, 96
- default relations, specifying, 139–42
- defining
 - dimensions, 85
 - hierarchies, 133
 - programs, 113
 - selections, 138
 - valuesets, 112
- deleting
 - database objects, 118
 - hierarchies, 138
- demonstration databases, 36
- Description property, 78, 94
 - dimension, 87
 - measure, 95
- DescriptionExp property, 94
 - dimension, 87
 - measure, 95
- DescriptionPlural property, 87
- DescriptionPluralExp property, 87
- detaching databases, 82
- Dictionary Menu, displaying as
 - popup, 46
- dimension
 - defining a hierarchy for a, 53
 - fastest varying, 158
 - how to add values, 54
 - how to define a, 52
 - relation, 9
 - slowest varying, 158
 - values, 8
- dimension status
 - adding or removing values, 151
 - defined, 150
 - resetting to all values, 153
- dimension values
 - as report labels, 6
 - how to delete from composites, 184
 - order, 131
 - used as constants, 218
- Dimensioned By property
 - dimension, 88
 - measure, 94
- dimensions, 3
 - aggregate, more and less, 9
 - choosing, 3
 - conjoint, 237
 - Data Dictionary structures, 86
 - data types, 5
 - default relations, 139–42
 - defining, 85
 - defining selections, 138
 - deleting hierarchies, 138
 - described, 85
 - editing, 85
 - hiding, 126
 - hierarchical, 56, 128
 - installing, 86
 - labels for, 85
 - level of detail, 4
 - properties, 87
 - setting up hierarchies, 133
 - sorting values, 137
 - uninstalling, 128
- DimensionType property, 88
- disconnecting from Express, 18
- domain authentication
 - when connecting to Express, 25
- drilling, 41

E

Edit Window, 114
editing
 dimensions, 85
 programs, 114
 valuesets, 112
EIF files
 exporting data, 108
 importing data, 106–8
embedded-total dimensions
 described, 130
 setting up, 133
errors
 compilation, fixing, 115
evaluating expressions
 changing the default behavior, 178
 default behavior, 178
Exception tools
 Measure option, 226
 Numeric option, 226
executing Express commands, 100, 149
executing programs, 115
exporting data
 status, 109
 to EIF files, 108
Express
 connecting, 17
 connecting to Express Server, 19
 disconnecting, 18
 programming language, 32
 running on same machine as Administrator, 28
 running on same machine as Personal Express, 29
 supporting multiple connections, 24
Express 5.x database
 upgrading to Express 6.0, 70

Express Client products
 defined, 17
Express command dialog box, 149
Express commands
 =, 177
 ALLSTAT, 153
 CHGDFN SEGWIDTH, 181
 CHGDFN with COMPOSITE
 keyword, 241
 CHGDFN with DIMENSION
 keyword, 242
 DEFINE COMPOSITE, 173
 executing, 149
 executing in Administrator, 100
 FILEREAD, 180
 LIMIT, 149
 MAINTAIN, 184
 REPORT, 186
 ROLLUP, 61
 STATUS, 151
Express data
 exporting, 108
 importing, 106–8
Express database
 importing data from, 107
Express Server
 applications, 2
 memory usage, 190
expression
 source, 177
expressions
 defined, 11
extension files
 of multifile databases, 192

F

fastest varying dimension
 illustrated, 158

FILEREAD command, 180
 maintaining conjoint dimensions,
 239
files
 Administrator INI, 21
 client application INI, 21
 database, needed by Administrator,
 35
 multifile databases, 192
 ruled, 104
 structured, 104
 supporting database, 20
 WIN.INI, 97
 XADMIN.INI, 18
 XPDDDATA.DB, 35, 123
finding
 database objects, 121
fixing compilation errors, 115
foreign keys, 13
FormatFormula property, 94
formulas
 data-driven, 93
 defined, 11, 89
 saving space in database files, 157
 setting number format, 91
 setting number format defaults, 96
 uninstalling, 128

G

getting help, 46

H

Help
 context-sensitive, 47
 invoking, 46
hiding dimensions, 126

hierarchical dimensions
 adding, 134
 aggregations, 134
 calculating the hierarchy, 59
 creating and using, 56
 depth, 131
 embedded totals, 130, 133
 establishing parent-child relations,
 58
 finding the name of the parent
 relation, 61
 installing, 134
 levels, 129
 order of values, 137
 overview, 128–32
 parent relation, 135
 rolling up data for, 57
 setting up, 133
 using the ROLLUP command, 61
hierarchies
 deleting, 138
 descriptions, 135
 naming, 135
 STANDARD, 134
host authentication
 when connecting to Express, 26

I

ID Code property, 88
implicit relations, 10
importing data, 103
 from EIF files, 106–8
Inspector
 Advanced tab, 44
 closing, 43
 described, 43
 displaying, 43
 setting properties, 119

- Installed property
 - dimension, 88
 - measure, 94
- installing
 - Administrator, 228
 - databases, 77
 - dimensions, 86
 - hierarchies, 134
 - measures, 92
 - requirements, 227
- integrity
 - database, 8
- IsAttribute property, 88, 94

L

- language dimension
 - dangers of deleting values, 81
 - described, 80
 - providing name expressions, 87
- Language Dimension property, 78
- languages
 - current system, 79, 81
 - for databases, 80
 - supporting multiple, 81
 - with a client application, 79
- LD property
 - dimension, 88
 - measure, 94
- levels
 - described, 129
 - specifying, 136
- LIMIT command, 150
 - for valuesets, 111
- limiting access to data, 196
- locating
 - application databases, 84
 - data databases, 83
 - database objects, 121

- login programs, 217
- login temporary variables, 209, 218
- LongLabelVar property, 88

M

- main files
 - of multifile databases, 192
- main menu bar, 39
- Main Window, 37
- MAINTAIN command, 184
- maintaining
 - databases, 75
- Match tool, 225
- MDBSource property, 88, 94, 223
- measures
 - defined, 11
 - described, 89
 - installing, 92
 - property list, 93
 - setting number format, 91
 - setting number format defaults, 96
 - setting properties, 93
 - uninstalling, 128
- memory
 - efficient use of, 190
- menus, 39
- models
 - defined, 12
- modifying
 - databases, 75
- multidimensional data model, 13
- multidimensional databases
 - vs, relational databases, 12
- multifile database, 192

N

NA

- defined, 156

- NA Pages property, 94

- name expression properties

 - for dimensions, 87

- named composites

 - changing to unnamed, 175

 - converting to conjoint dimensions,
242

 - copying the definition, 175

 - creating, 173

 - defined, 172

 - getting information about, 175

- number format

 - defaults, 96

 - setting, 91

 - setting default, 97

 - storage string, 91

 - using defaults, 98

O

- OBJ function, 175

 - using to find the parent relation, 63

- objects

 - database, 2

 - database, defining, 51

 - how to view all, 50

- ObjectType property

 - dimension, 88

 - measure, 94

- ODBC data source

 - setting up for SQL reach-through,
208

- OLAP, 13

- OLTP, 13

 - one database in multiple files

 - saving, 192

 - order of dimension values

 - described, 131

 - specifying, 137

P

- pages

 - units of storage, 155

- parent relation

 - described, 130

 - specifying, 135

- password

 - database, 195

- PathName property, 78

- pattern of sparsity, 165

- performance

 - of loading and accessing data, 181

- permission

 - assigning to a dimension, 202

 - assigning to a dimensioned
variable, 199

 - assigning to an object definition,
197

 - creating for a database, 195

 - establishing permission conditions,
197

 - limiting access to data, 196

- PERMIT command, 196

- PERMIT_READ and

 - PERMIT_WRITE programs, 195,
196, 203

- pointers, mouse, 47

- popup menus, 46

- Post-Attach Program property, 78

- Post-Detach Program property, 78

- Pre-Attach Program property, 78

- Pre-Detach Program property, 78

primary keys, 13
 processing
 saving time, 154
 programs
 ASCII data loader, 105
 compiling, 114
 debugging, 116
 defined, 12
 editing, 114
 executing, 115
 fixing compilation errors, 115
 how to define, 113
 login, 217
 run at the client application
 shutdown, 142
 SQL data loader, 106
 properties
 database, 78
 defined, 43, 119
 dimension, 87
 measure, 93
 name, 95
 setting for database objects, 119
 setting for databases, 120

Q

QDR, 209, 218
 qualified data reference, 209, 218

R

random sparsity
 defined, 161
 ReadOnly property, 78
 rebuilding databases, 188
 relational database
 fetching data from, 95

importing data from, 106
 setting up an ODBC data source,
 208
 vs multidimensional database, 12
 relations
 between dimensions, 9
 defined, 89
 defining, 9
 implicit, 10
 multidimensional, 11
 multiple, 10
 naming convention, 11
 parent-child in a hierarchical
 dimension, 58
 setting number format, 91
 specifying default, 139–42
 remote data, 207
 connecting, 217
 login programs, 217
 security, 223
 Selector, 224
 source, 223
 RENAME command, 174
 renaming database objects, 117
 REPORT command, 186
 roll up
 defined, 57
 ROLLUP command, 61
 for sparse variables, 187
 rotating data, 90
 ruled files
 defined, 104

S

saved selections, *See* selections
 saving
 a database in multiple files, 192

- scoping
 - with the PERMIT command, 196
- searching
 - for database objects, 121
- security
 - connecting to Express Server, 20
 - for remote data, 223
- security options
 - connecting to Express, 25
- segment width
 - of variables, 181
- selections
 - list type, 138
 - overview, 138
 - script type, 139
- Selector and remote data, 224
- server connections
 - security options, 25
 - specifying one or more, 23
 - specifying the Express Server version, 23
 - supporting multiple, 24
- setting number format, 91
 - defaults, 97
- setting properties
 - for database objects, 119
 - for databases, 78
- ShortLabelVar property, 88
- ShortName property, 94
 - dimension, 88
 - measure, 95
- ShortNameExp property, 94
 - dimension, 88
 - measure, 95
- ShortNamePlural property, 88
- ShortNamePluralExp property, 88
- siblings
 - specifying order, 137
 - term defined, 131
- slice of data, 4
- slowest varying dimension
 - illustrated, 158
- Sort tool, 226
- source expression
 - defined, 177
- source for remote data, 223
- sparse data
 - defined, 158
- sparse variables
 - assigning data to, 177
 - creating a, 162
 - defined, 162
 - rolling up data, 187
 - selecting values, 185
 - using Express commands to define, 171
- sparsity
 - controlled, 159
 - of data, 158
 - random, 161
- sparsity pattern
 - defined, 165
- SQL
 - definition, 13
- SQL data
 - importing, 106
- SQL databases
 - connecting to, 95
- SQL reach-through
 - setting up an ODBC data source, 208
- STANDARD hierarchy, 134
- starting Administrator, 36
- status
 - of dimensions, defined, 150
- status bar, 47
- STATUS command, 151
- StorageType property, 94

structured files
defined, 104

T

target variable
defined, 177
temporary
variables, 157
time dimensions
data types, 5
implicit relations between, 10
toolbar, 40
Tools
Attribute, 225
exception, 226
Match, 225
Sort, 226
Top/bottom, 226
Top/bottom tool, 226

U

uninstalling
databases, 127
dimensions, 128
measures, 128
unnamed composites
converting to conjoint dimensions,
242
defined, 171
naming, 174
upgrading an Express 5.x database,
70
User Data property, 78
UserVisible property
dimension, 88
measure, 94

V

values
dimension, 8
dimension, adding, 54
valuesets
defined, 12
editing, 112
how to define, 112
variables
data, 13
defined, 7, 89
determining segment size, 181
editing, 67
entering data in a, 67
how stored, 155
how to define, 65
saving data from a formula, 156
setting number format, 91
setting number format defaults, 96
sparse, 162
sparse data, 158
target, 177
temporary, 157
temporary for login, 209, 218
that share composites, 165
uninstalling, 128
varying dimensions
illustrated, 158

W

Width property, 88, 94
WIN.INI file, 81, 97
worksheets
defined, 12

X

XADMIN.INI file, 18

XCA, 217

Xn prefix, 86

XPDDDATA.DB

configured database information,
123

defined, 35

XXX.LANGDIM dimension, 80

