Oreview.com
The Independent Voice of Oracle Product Users

APPLICATION DEVELOPMENT

William G. Brown
OTJ, Winter 1996

Oracle Financial Analyzer: A Look Under the Hood

In-depth discussions of OFA's architecture and how to specify information so your system is easy to develop and maintain.

Oracle entered the growing OLAP market last year with its acquisition of a multidimensional database and a set of applications from Information Resources Inc. (IRI). One of the applications, Oracle Financial Analyzer (OFA), is a powerful application that uses the Express multidimensional database to analyze financial data. OFA's unique tiered design lets groups within an organization's hierarchy customize the application to their own specific business requirements. OFA has been successfully used for budgeting, planning, financial reporting, product profitability analysis, sales forecasting, and activity-based costing in industries as diverse as banking, manufacturing, retail, consumer package goods, and telecommunications.

Most developers use the OFA front end to build their applications. In this article, I discuss how you can develop OFA applications faster and streamline ongoing maintenance by using the Express 4GL (fourth-generation language) to access the Express database layer in OFA. Specifically, I describe OFA's architecture and how to specify key information so your system is easier to develop and maintain. I review the key Express Server objects that OFA maintains, multiuser development with OFA, how to maintain these objects at the database layer, and how to use Express to load external data sources. I also give you some suggestions on how to debug OFA problems.

OFA Architecture

A typical OFA application consists of a PC-based DBA workstation that controls a shared Express database. The PC DBA workstation has a copy of the OFA front end and the Personal Express database containing OFA application code. (See Figure 1.) Using

tools provided by OFA to define various OFA objects (such as dimensions, hierarchies, and hypercubes), a DBA customizes OFA for a specific application on the DBA workstation. The DBA then distributes the OFA objects to the shared Express database on the server. User workstations access the shared database, which acts as the central repository for catalogs and data. This arrangement lets a DBA test changes on the PC DBA database before applying them to the shared database on the server accessible by all users.

How OFA Interacts with Express

The key to simplifying development and ongoing maintenance in OFA is understanding how OFA interacts with Express. Whenever a DBA creates an object using the OFA front end, OFA automatically defines that object and a set of related objects in Express. The related OFA-created objects include valuesets, variables, and relations that control how OFA behaves. In addition, OFA makes entries into its own meta-dictionary, implemented as a series of catalogs. Catalogs are Express variables that store application information. Understanding the objects OFA creates, how to interpret OFA catalogs, and how to change the catalogs if needed will greatly improve your productivity as an OFA developer.

Until you are an expert in how OFA objects and their catalogs interact with one another, it is best to create and maintain objects in OFA using the OFA front end only. Once you're familiar with the object relationships that OFA creates, you can use Express to add and maintain objects directly in the catalogs, bypassing the OFA front end altogether. The simplest way to begin to understand the nature of these relationships is to add objects using the OFA front end so you can see the relationships that OFA creates. After you add objects and are able to see these relationships, whenever you change a catalog value you will know which associated objects need to be changed as well. You must rename objects before distributing them to the user community; therefore, it is important that you choose object names carefully before you begin creating your OFA objects.

Although using Express can speed up development of your OFA application, many changes are not immediately recognized by OFA until you either restart the OFA front end or register the changes with the OFA front end (which is called re-registering OFA). Re-registering forces OFA to scan the critical database information and initialize the OFA front-end arrays. The simplest, least graceful way to re-register is to restart the OFA front end. Oracle support has a technical note entitled "LC.CATALOG and Setting Up Customization in OFA (FMS 4.6)" that describe how to re-register without restarting the OFA front end.

Important OFA Objects

As you work with OFA, you will find that you spend most of your time working with the dimension and hierarchy objects. Each OFA-created object associated with a dimension

contains a prefix of up to six characters that is used as an abbreviation for the dimension. (For example, you might assign the prefix "ORG" to the ORGANIZATION dimension in an application.) Dimension prefixes are important because they can help you understand the Express structures that OFA defines. Getting acquainted with the prefix values facilitates your understanding of the OFA database and can help you create intelligent catalog entries.

To find a dimension prefix, look in the OFA front end or in the dimension catalog (DM.CATALOG). Table 1 lists the standard OFA-created dimension objects. You can use Table 1 as a checklist when developing your application to ensure that you are populating your tables correctly. Keep in mind that once you create a dimension setting for an object name, prefix, dimension type, or width, you cannot change it without substantial effort. Therefore, it is important that you determine the dimension settings carefully before creating the dimensions using the OFA front end.

For every new dimension you create, OFA also creates objects that store and control hierarchies. Each OFA-created object associated with a hierarchy ends with a prefix of up to six characters that is used as an abbreviation for the hierarchy. Table 2 lists the hierarchy objects created by OFA. OFA uses the hierarchies you define to determine the order of the dimension members in dialog prompts. If you do not have a hierarchy or are not using one, the dimension members are shown in the order you created them.

The OFA Catalogs

OFA catalogs contain information about application objects. Each OFA catalog variable is defined by two dimensions: an ENTRY dimension and a PROP (property) dimension. The ENTRY dimension is the list of objects added to the database by a DBA or end user. The PROP dimension is the list of properties (for example, who created the object and references to other objects). OFA associates a number of application objects with the ENTRY dimension. Most of these objects either are temporary (their values disappear when you exit OFA) or are filled in dynamically by OFA. These objects will rarely, if ever, have any impact on the development of your OFA application.

Each OFA-created object associated with a catalog begins with a prefix that is used as an abbreviation for the catalog. Table 3 lists the most common OFA catalogs and their prefixes. You will also find it useful to remember that catalogs are text variables. In Express, text variables have an important feature that is easily overlooked, even by experienced Express programmers. A single cell can contain many lines of text data. Use the TABLE command to view the catalogs and then press the F4 key twice to view the complete text of a catalog variable cell.

Each catalog has a different set of properties associated with it. Table 4 lists the properties common to each catalog, as well as the properties specific to the attribute, hierarchy, and model catalogs. If you are using the OFA front end to create your objects,

OFA will automatically populate the catalogs with the relevant properties. If you are using Express, you must populate the catalogs with the relevant properties yourself. In addition to knowing the properties of each catalog, you should understand the functions of the attribute, hierarchy, and model catalogs.

It is especially important to use the OFA front end to maintain the dimension and financial data catalogs (DM.CATALOG and FD.CATALOG). OFA automatically adds and deletes OFA-created objects whenever you change these catalogs.

Example of OFA's Interaction with Express

Using the creation of a dimension as an example, you can see how OFA interacts with the Express database. Assume that we have created a dimension called ORGANIZATION and have also added a corporate rollup hierarchy to ORGANIZATION to be named CORP_ROLL. Following are the most important values in this example's dimension catalog (DM.CATALOG):

	DM.CATALOG
DM.PROP	DM.ENTRY
	ORGANIZATION
CLASS	PERSONAL
MODIFIER	AA
TIME.MODIFIED	96/09/06 14:02:08
DESCRIPTION	ORG.DESC
PREFIX	ORG

Notice that the DM.CATALOG has a DM.ENTRY titled "ORGANIZATION." The values of the catalog reflect the screen inputs required when defining this dimension. Of particular importance is the property PREFIX, which is used by OFA when creating the related objects for this dimension.

When we defined the dimension ORGANIZATION, OFA created a set of related Express database objects. The following are the most important objects for an OFA developer: ORG.DESC, ORG.LBL.ROW, ORG.LBL.COL, ORG.STD, FMSHDIM.ORG, FMSHREL.ORG, FMSHDEP.ORG, and FMSHSEQ.ORG. Refer to Tables 1 and 2 for more information on how these objects are used by OFA.

Following are the most important values in our example's hierarchy catalog (hi.catalog):

	HI.CATALOG
HI.PROP	HI.ENTRY

	HI.AA12345
CLASS	PERSONAL
MODIFIER	AA
TIME.MODIFIED	96/09/06 14:03:00
DIMENSION	ORGANIZATION
TYPE	TREE

Even though we gave the hierarchy the name "CORP_ROLL"in the OFA front end, OFA used an internal name for this hierarchy. This name is a concatenation of the object's prefix (HI for hierarchy), a period, the user's unique internal code (AA for the DBA), and a randomly generated number (12345 in this example). Note that OFA also added a member to the FMSHDIM.ORG dimension with the same name, HI.AA12345. OFA uses internal names for hierarchies, models, and attributes. In the next section I show an example of how to change this internal name to one that is both easier to remember and self-documenting.

Maintaining Catalogs Behind the Scenes

The primary reason you should maintaining OFA catalogs yourself is that you can then create more readable and understandable names for the Express objects that OFA creates. With the exception of the object names for dimensions and financial data items, you cannot specify the names for the objects created using the OFA front end. As you add hierarchies, models, and attributes, OFA automatically creates names for the catalogs and the objects they describe. As you saw previously, the OFA front end might create the name HI.AA12345 for the Corporate Rollup for the ORGANIZATION dimension. I recommend renaming hierarchy, attribute, and model objects to more readable names. Wherever possible, rename them so that they conform to the OFA standard of embedding a prefix in the object name that references the relevant dimension.

Hierarchies control the aggregation of data along a dimension. When a new hierarchy is created, OFA adds a new member to the hierarchy dimension HI.ENTRY and fills in the hierarchy catalog (HI.CATALOG). However, OFA does not define a new object; instead, it adds a member to the FMSHDIM.prefix dimension. (Refer to Table 2.) The relationship between the HI.CATALOG and FMSHDIM.prefix assumes that the members are identical. If you change the hierarchy name, you must rename the members of both objects. In the hierarchy example given previously, we want to rename the hierarchy HI.AA12345 to CORP_ROLL. To do this, you must change the name of the hierarchy in both the HI.ENTRY dimension and the FMSHDIM.ORG dimension as follows:

```
maintain HI.ENTRY rename
'HI.AA12345' 'CORP_ROLL'
maintain FMSHREL.ORG rename
'HI.AA12345' 'CORP_ROLL'
```

Attributes control the relationships between dimensions. For example, your OFA application may receive budget data stated in different currencies from different countries. An attribute would relate each COUNTRY dimension to a CURRENCY dimension to facilitate translating the currency to U.S. dollars. Assume that the dimension prefixes are CTRY and CURR, respectively. When a new attribute is created, OFA adds a new member to the attribute catalog dimension RL.ENTRY, fills in the attribute catalog (RL.CATALOG), and creates a new object. Assume that OFA has named the new attribute dimension member RL.AA54321 and the new object .RL.AA54321R. To change the OFA-created object names to names based on the dimension prefixes, you must modify both RL.ENTRY and RL.CATALOG as follows:

```
maintain RL.ENTRY rename
'RL.AA54321' to 'CTRY_CURR'
rename .RL.AA54321 CTRY_CURR
limit RL.ENTRY to 'CTRY_CURR'
limit RL.PROP TO 'OBJ.NAME'
RL.CATALOG='CTRY_CURR'
```

Models control the calculation of data along a dimension. Although hierarchies will only aggregate, models can perform a wide range of calculations. When you create a new model, OFA adds a new member to the model dimension MD.ENTRY, fills in the model catalog (MD.CATALOG), and defines the object indicated by the value EQU.VARIABLE in the MD.CATALOG. If you change the object name yourself, you must modify both MD.ENTRY and MD.CATALOG as follows:

```
maintain MD.ENTRY rename
'MD.AA12345' 'HC_BUDGET'
rename .MD.AA12345E HC_BUDGET_T
limit MD.ENTRY to 'HC_BUDGET'
limit MD.PROP to 'EQU.VARIABLE'
MD.CATALOG='HC_BUDGET_T'
```

Changes to the model must be applied to the object indicated by the value of EQU.VARIABLE. Once you've made the model changes, you must use either the OFA front end or Express to compile the model so that the changes will be reflected. Based on the value for MD.ENTRY, OFA will automatically create PROGRAM and RECALC objects if they are required. Use the program MD.COMPILER to compile models. To compile the model in the above example, issue the command:

```
call md.compiler('HC_BUDGET')
```

Loading Data and Dimension Values into OFA

Populating OFA objects with data is usually done by extracting data from existing production systems such as the General Ledger, Payroll, or Sales System. Extracts from

these systems will also provide the master dimension values and rollup structures needed to populate OFA dimensions and hierarchies.

If the source for your OFA application is the Oracle General Ledger, you should use the built-in Oracle General Ledger interface to load data and dimension values. To load data, dimensions, and hierarchies into OFA from external sources other than the Oracle General Ledger, you will need to use the Express 4GL. (See Listing 1.)

When designing your data loader, keep in mind that OFA assumes that the flow of changes is from the PC DBA workstation to the server. If you want to load dimensions and hierarchies on the server, you need to update the PC DBA database with the dimension and hierarchy changes from the server. The process to update the PC DBA will vary with each OFA installation.

Important: If you are loading a hierarchy from an external file, you must calculate the values for the OFA depth and sequence variables yourself. (A shareware copy of a utility that calculates the values for depth and sequence objects based on the values in the FMSHREL.prefix is available on the *OTJ* Web site at http://www.otj.com.) If you fail to do this, the objects will not be calculated and you will not be able to use your hierarchy. See Listing 2 in the electronic version of this article on the OTJ Web site for a simple example of how to load a hierarchy.

Multi-User Development

OFA stores all development catalogs and their associated objects on the PC DBA workstation. Once you and your colleagues are familiar with the OFA catalogs, their associated objects, and native Express, you can set up an OFA development environment that lets multiple developers work simultaneously on different parts of your OFA application. If you are developing on a PC workstation that is not the PC DBA workstation, you can simply move your finished code to the PC DBA workstation using the Express EXPORT command. The EXPORT command builds an Express Interchange Format (EIF) file that a DBA may IMPORT to his/her PC workstation. This process will vary depending on the specific OFA objects (hierarchies, dimensions, models, reports, and so on) that you are maintaining. A brief example of how you would maintain a model follows.

Assume you are developing on a PC workstation other than the PC DBA workstation. You have just finished the Headcount Budget model that is run against the LINE dimension. The model dimension (MD.ENTRY) member is HC_LINE. The object indicated by the property dimension (MD.PROP) member EQU.VARIABLE in the model catalog (MD.CATALOG) is HC_MODEL_T. You will need to move these objects to the PC DBA. The objects indicated by property dimension (MD.PROP) members PROGRAM and RECALC will be regenerated when the DBA compiles the model. The commands below use an OFA utility to compile the model. This example assumes that

you did not create any new LINE dimension members. You would issue the following commands:

```
limit MD.ENTRY to 'HC_BUDGET'
limit MD.PROP to all
limit LINE to all
export MD.CATALOG HC_MODEL_T to
eif file 'HCMODEL.EIF'
```

The DBA would then issue these commands and re-register the changes:

```
import all from eif file
'HCMODEL.EIF'
call md.compiler('HC_MODEL')
```

Debugging OFA

With the information in this article, you should be able to use Express to speed up your OFA application development and maintenance - but this is just a starting point for gaining an expert level of understanding about OFA. You can also use Express as a debugging aid as follows.

Set the Personal Express option IFCOPY to YES after starting OFA. Set PAGEPAUSE to NO. (If you forget to set PAGEPAUSE to NO, Express will pause after one screen is filled with output. OFA will then signal an error message and require that the session be restarted.) IFCOPY will echo on your screen the program calls between the OFA front end and the Express database. Pay particular attention to anything containing the word CATALOG or any of the prefixes listed in Table 1. This information can help you pinpoint problems and learn more about how OFA behaves.

If you are using the TABLE command to view objects, be sure to escape out of the table before switching back to the OFA front end.

Leveraging Your OFA Application

Learning how to access the Express database layer in OFA in the early stages of your OFA implementation will enable you to tie OFA to your existing production systems, increase developer productivity, and streamline ongoing OFA application maintenance. By following the tips and techniques summarized in this article, you will be well on your way to taking full advantage of the insights OFA can provide into your financial applications.

William G. Brown is a consultant with Symmetry Corp., an OLAP consulting firm headquartered in San Rafael, Calif. William has six years of experience implementing OLAP applications at Fortune 500 companies, including Oracle Corp.'s own OFA-based financial analysis and budgeting system. You can email William at wgbrown@symcorp.com.

FIGURE 1

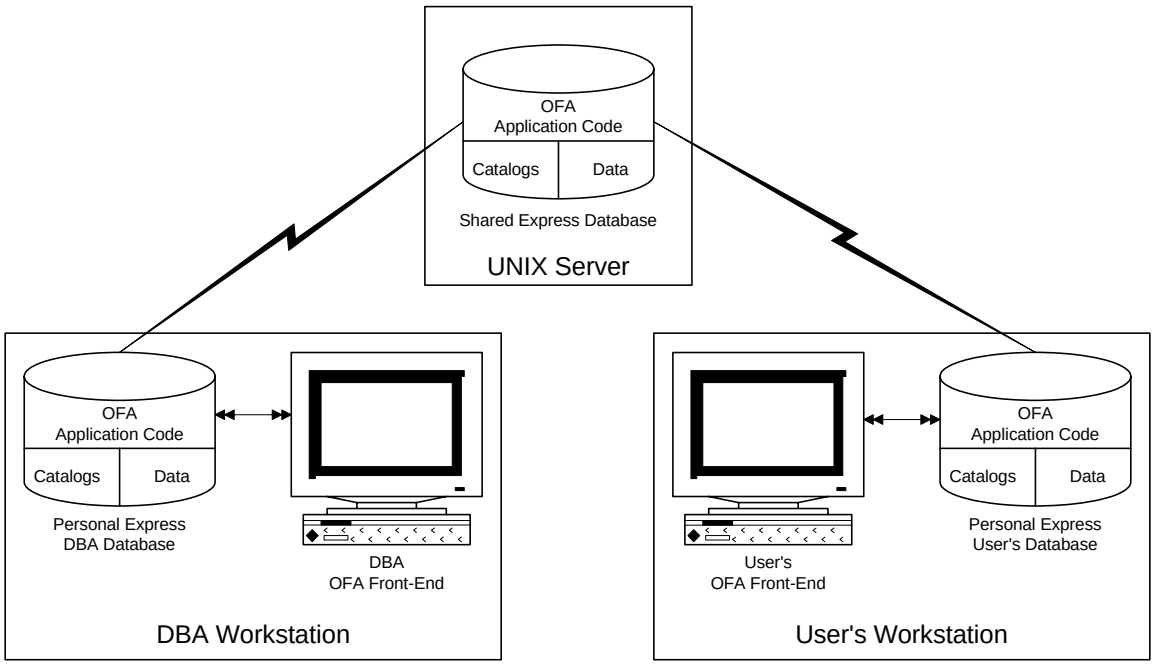

Figure 1: OFA Architecture

TABLE 1. Objects Created by OFA for Dimensions

Note: The word prefix denotes any dimension created with OFA.	
prefix.DESC	Stores the descriptive values for dimension members.
prefix.LBL.ROW	Stores the long descriptive values for dimension members.
prefix.STD	Boolean variable that indicates whether the dimension member was distributed from the DBA or was a locally created member. If the variable is set to NO, OFA assumes it was locally created. If it is set to YES, then OFA assumes the same dimension member exists in the server database and allows the data to be uploaded or downloaded.
prefix.AGG	Controls the method OFA uses to aggregate data over time. This object exists only if the dimension supports time aggregations. To modify this setting, see the Maintain Dimension option in OFA. You can use one of the following six methods to control time aggregation:
Internal Value	**Meaning**
SUM	Adds months into quarters (default)
BEG	Moves first month only into quarter
END	Moves last month only into quarter
AVG	Averages months into quarter
AVGC	Averages across ORG and TIME
RECALC	Recalculates dimension members after the data have been aggregated over time. Used to calculate ratios correctly at the quarter and year.
prefix.BW	An integer variable that can be used to control better/worse variance calculations. This object exists only if the dimension supports better/worse variance indicators. To modify this setting, see the Maintain Dimension option in OFA. You can use one of the following two settings for this option:
Internal Value	Meaning
1	Subtract Actuals from Budget
-1	Subtract Budget from Actuals **Important**: It is the developer's responsibility to make use of this variable when constructing variance calculations. The example below shows the variance calculation that would be entered in the **Maintain Financial Data** form option. (ACTUALS-BUDGET)*nafill(ACCT.BW 1) The nafill function is used here to set the default behavior of the variance calculation. If the user has not filled in all the values of the better/worse variable, nafill ensures that the default behavior of the variance calculation is ACTUALS-BUDGET.

TABLE 2. Objects Created by OFA for Hierarchies

Note: The word prefix denotes any dimension created with OFA.	
FMSHDIM.prefix	Lists the hierarchies for the dimension. Every hierarchy created for the dimension has a member in this list as well as in HI.ENTRY.
FMSHREL.prefix	Stores the parent/child relationship used to calculate totals and control drilling into more detailed data. This structure is often referred to as a self-relation because it relates the dimension to itself.
FMSHDEP.prefix	An integer variable that stores dimension members' depth in the hierarchy. Depth is a count of the number of members between a given member and the top of the hierarchy. The depth at the top of a hierarchy is zero.
FMSHSEQ.prefix	An integer variable that stores dimension members' sequence in the hierarchy. Sequence is the order in which the members appear in the hierarchy.

TABLE 3. Common OFA Catalog Prefixes

Note: Catalogs are named prefix.CATALOG. Catalog dimensions are named prefix.ENTRY and prefix.PROP, respectively. For example, the Master Catalog is named MC.CATALOG, with dimensions of MC.ENTRY and MC.PROP.

Prefix	Object Catalog	Object Function
MC	Master Catalog	Catalog of major system catalogs
DM	Dimensions	All dimensions created via the OFA front end
HI	Hierarchies	Dimension hierarchies
RL	Attributes	Relation of two dimensions
FD	Financial Data Items	Hypercubes of data, both stored and calculated
RE	Reports	User-created reports
BR	Folders	Lists of reports, worksheets, or graphs
BW	Worksheets	User-created worksheets
SO	Solve	Rules to calculate a hypercube's derived data using hierarchies and models
GR	Graphs	User-created graphs
DI	Display	Displays formatting for reports, worksheets, or graphs
SG	Segment	Supporting object for reports, worksheets, or graphs
CO	Component	Supporting object for reports, worksheets, or graphs
SL	Selection	Dimension selection for reports, worksheets, or graphs
DB	Database	Names and locations for OFA databases

TABLE 4. Catalog Properties

Properties Common to All Catalogs	
Property	**Description**
CLASS	PERSONAL (Locally created) or STD (Distributed from the DBA. The user may not change the item.)
MODIFIER	User ID code of the object owner
TIME.MODIFIED	Date and time stamp of last update
DESCRIPTION	Text description of object (not used in most cases)
Attribute Catalog (RL.CATALOG) Properties	
Property	**Description**
REL.TYPE	Type of relation ONE to MANY or MANY to MANY
BASE.DIM	Base dimension for relation
AGGR.DIM	Group dimension for relation
OBJ.NAME	Express name for object that stores the relation
Hierarchy Catalog (HI.CATALOG) Properties	
Note: Other properties not listed here are used with the Oracle GL interface.	
Property	**Description**
DIMENSION	Dimension the hierarchy is based on
TYPE	Always TREE
Model Catalog (MD.CATALOG) Properties	
Property	**Description**
BASE.DIM	Dimension the model runs against
COMPILE.NEEDED	Shows whether the model has been compiled
VARIABLES	Variables the model can run against
DIMENSIONS	All dimensions used in the model. Contains the value in BASE.DIM and, if time functions such as LAG and LEAD are used, TIME.
PROGRAM	Name of the compiled version of the model
RECALC	Name of the compiled version of the model for recalculated lines only. See the section on dimension objects, specifically: prefix.AGG
EQU.VARIABLE	Text variable where changes to the model are made

LISTING 1. Example of Loading an External Data File

```
DEFINE READ_GL PROGRAM
LD Read GL level data
PROGRAM
"|-------------------------------------------------
------------------
"|
"| Program: READ_GL
"|
"| Purpose: Reads the supplied GL flat files,
maintains dimensions
"|          and populates the actuals hypercube.
"|          The month of the data file is stored in
the header record.
"|          O (for ORG) and A (for account) are
appended to the front of
"|          each dimension member. OFA requires that
dimension
"|          members begin with a character.
"|
"|-------------------------------------------------
-------------------
"| Written By:  William Brown   (Symmetry)
"|
"| Changed By:  Date:           Change:
"|
"|-------------------------------------------------
-------------------

variable _fid     integer          "File identifier
used by the FILEOPEN command

variable _month   text           "The month of data
contained in the in file.

"Populated by the first record in the file.


trap on error                          "Set error
handling logic on just in case
```

```
pushlevel 'READ_GL'                "Save the status of
the dimensions
push ACCT ORG TIME

"This program makes two passes through the data
files. The first pass maintains
"(adds) any new dimension members. The second pass
reads the ACTUAL data. Adding
"dimension members first and then reading data makes
the database more efficient.

_fid = fileopen('/data/DATAFILE.TXT') read)
                                    "open the datafile
to be read

"Skip the first record. The first record lists the
month of data that the file
"contains. It is only required when data, not
dimensions, is being read in.

fileread _fid stopafter 1

"After skipping the first record process the rest of
the file adding dimension members.

fileread _fid -
     col 1  w 10 append lset 'O' ORG -
     col 12 w 15 append lset 'A' ACCT   -

fileclose _fid                      "Close the file
and then reopen it to restart the
                                          "process at
the top of the file

update                               "Save the changes

_fid = fileopen('/data/DATAFILE.TXT') read)

"Read the first record which holds the month of data
in the file

fileread _fid stopafter 1 col 1 w 5 _month

limit TIME to _month
```

```
"Select the month of data that is being read. The
value in the first record has
"all ready been formatted to the OFA specifications.

"Now that the dimensions have been populated from the
first pass through the file
"the second pass does not add any dimension members
but populates the ACTUAL cube.

fileread _fid -
     col 1  w 10 match lset 'O' ORG -
     col 12 w 15 match lset 'A' ACCT -
     col 30 w 15 ACTUAL

update                                    "Save the
changes

fileclose _fid                           "Close the
data file
poplevel 'READ_GL'                  "Restore status
return                                    "Return to
calling program

error:                                      "Error
processing section
signal errorname errortext      "Signal error to
calling program
poplevel 'READ_GL'                  "Restore status
fileclose _fid                           "Close the
data file
```

LISTING 2. Example of Loading a Hierarchy

```
DEFINE READ_ACCT PROGRAM
LD Read account file, hierarchy
PROGRAM
"|-------------------------------------
"|
"| Name: READ_ACCT
"|
"| Purpose: Reads the ACCOUNT hierarchy files in the
data directory and maintains
"| the ACCOUNT dimension. Reads descriptions,
hierarchies and sets OFA objects
"| correctly.  This program assumes only one
hierarchy for ACCOUNT.
"| ACCOUNT prefix: ACCT
"|
"| Written by: WGBROWN   (SYMMETRY)
"|
"| Changed By: Date      Change
"|
"|
"|-------------------------------------

variable _fid integer                    "file
identifier used by the FILEOPEN command

trap on error                               "Set
error handling logic on just in case

limit FMSHDIM.ACCT to 'MGMT_ACCT'        "Select the
hierarchy being populated
_fid = fileopen('/data/ACCOUNT.TXT') read)

"Open the file to be read.

"The first pass through the file adds any new ACCOUNT
dimension values.
"The first column (col 1 w 15) holds the child value
and the second column listed
```

```
"(col 80 w 15 ) holds the parent value.  Any member
that is a parent should also be
"in the child column.  Add an 'A' to the dimension
value because OFA requires that
"dimension members start with an alpha character.

fileread _fid -
   col 1 w  15 append lset 'A' ACCT -
   col 80 w 15  append lset 'A' ACCT -

fileclose _fid


fileread _fid -
   col 1 w  15 match lset 'A' ACCT -
   col 16 w 10  ACCT.LBL.COL -           "populates
description variables
   col 27 w 30 ACCT.LBL.ROW -
   col 58 w 30 ACCT.DESC  -
   col 99 w  1 ACCT_TYPE -                 "asset,
liability, revenue, expense, hc
   col 101 w  15 lset 'A' FMSHREL.ACCT   "hierarchy
value, parent of col 1

fileclose _fid
"close file

upd

limit ACCT to all
"Set the aggregation method for each dimension member
"Account type of A or L get averaged over time
otherwise they get summed.
ACCT.AGG= if ACCT_TYPE eq 'A' or ACCT_TYPE eq 'L'
then 'AVG' ELSE 'SUM'

return

error:
"error processing
fileclose _fid
SIGNAL ERRORNAME ERRORTEXT
END
```