

# Oracle Hyperion Financial Management Tips & Techniques

Design, Implementation & Support

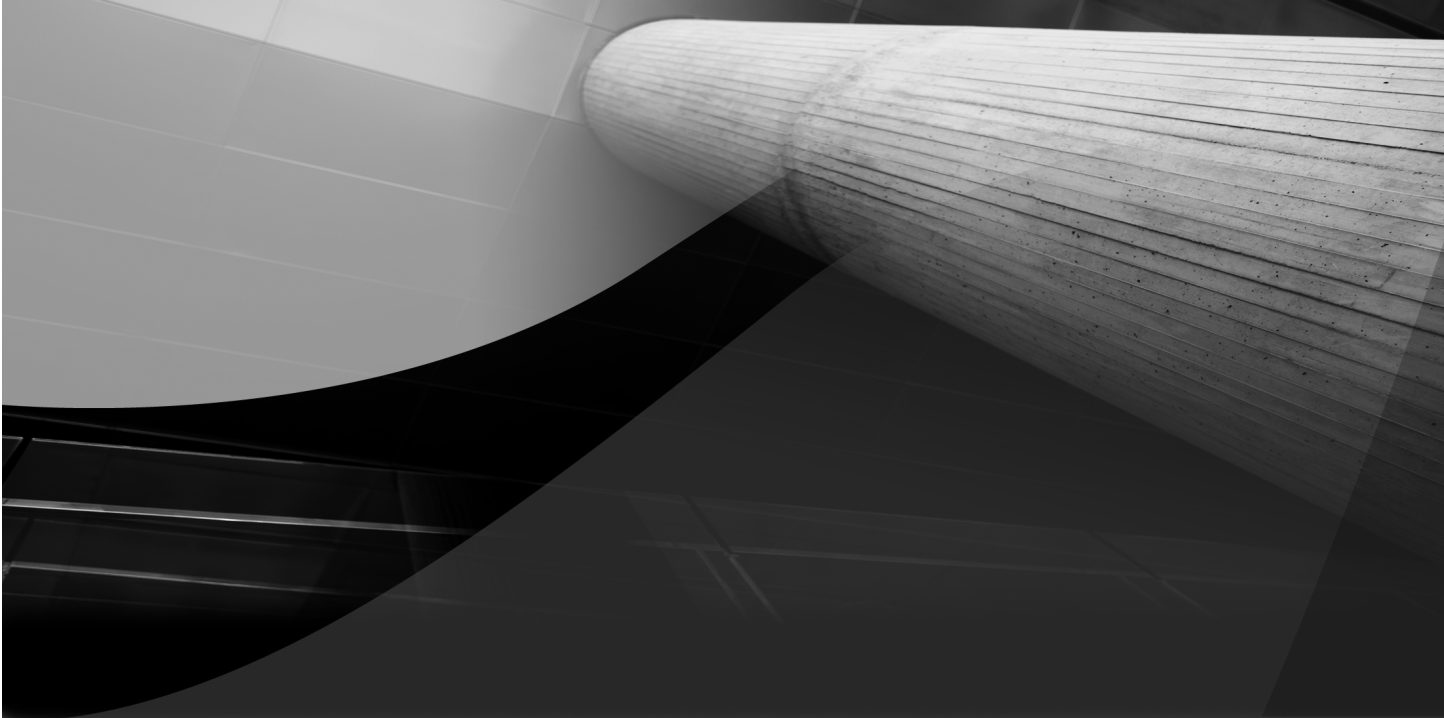
ORACLE®  
HYPERION

**Peter John Fugere, Jr.**

Oracle ACE and Vice President at Edgewater Ranzal







# CHAPTER 1

## Designing Your Oracle HFM Application

## 2 Oracle Hyperion Financial Management Tips & Techniques



After hours of sales demonstrations and information about the products, you are ready to start implementing Oracle Hyperion Financial Management (HFM). The work may seem a bit overwhelming at first, but this chapter will help. In this chapter, we will cover the basics of the full suite of products, implementation strategy, identifying your internal team and consultation partner, what a good timeline would look like, and—most critical—designing the HFM application. A good foundation built during the design ensures success. With these tools you should be ready to take on implementing HFM.

### Enterprise Performance Management

Enterprise Performance Management (EPM) is a set of analytic processes and tools that allow a business to identify, track, measure, and achieve goals. Within the EPM group there are several domains. Each domain focuses on a key business process or function. The Oracle tools focus on the most common EPM domains, which include:

- Planning and Forecasting
- Financial Management
- Strategy Formulation
- Supply Chain Effectiveness

The tools Oracle offers are a full suite of products that focus on each process individually or in concert, and they provide the capabilities that management needs to meet its goals. HFM is part of this suite of tools.

### The Oracle EPM Suite Overview

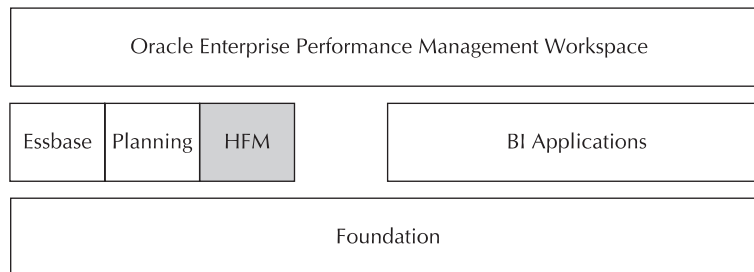
Several products make up the Oracle Enterprise Performance Management (EPM) suite. At the core of the suite are three products: Oracle Hyperion Financial Management, Essbase, and Planning. Each product, while designed and built for each of the key EPM domains, also works in concert to integrate the entire process. For example, Financial Management is

exceptional for computing complex currency translations, allocations, eliminations, and consolidations. Planning is built for driver-based planning, and managing multiple scenarios such as forecasting and financial business modeling. Essbase is a powerful analytic tool that can handle large volumes of data.

As you can see in Figure 1-1, each core product fills a role between the web tier and the foundation. This is not to say that there is no overlap between these tools. I have seen Essbase applications that include consolidation and elimination. I have worked with HFM applications that have planning, budgeting, and forecasting functionality built into them. However, each product is exceptional for its own focused task.

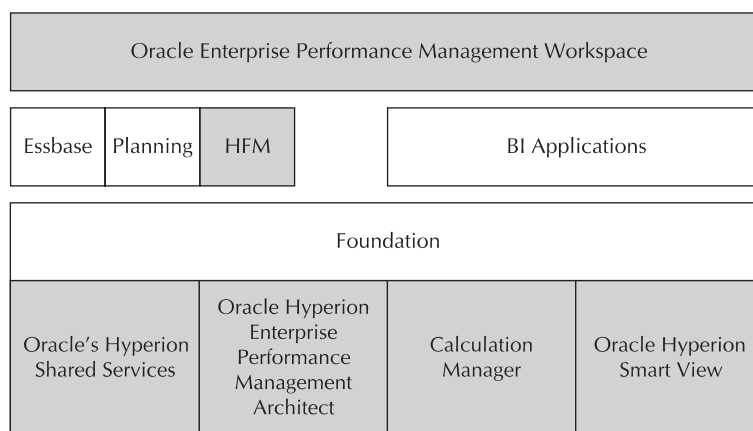
The three core products all sit on top of the Oracle Foundation. Figure 1-2 shows all of the services included in the Foundation. The core parts I am going to discuss are

- **Oracle Hyperion Shared Services** Shared Services provides a central location to manage user provisioning, lifecycle management, and task flow management for all EPM System products. Also Shared Services includes the Shared Services Registry, which is a central repository that stores and reuses information for most EPM system products installed.
- **Enterprise Performance Management Architect (EPMA)** EPMA allows creating, maintaining, and synchronizing the core applications through a graphical user interface. Enterprise Performance Management Architect works with: Calculation Manager, Planning, Financial Management, Essbase, Profitability, and Cost Management.



**FIGURE 1-1.** *The full EPM suite*

## 4 Oracle Hyperion Financial Management Tips & Techniques



**FIGURE 1-2.** *The Foundation Services*

- **Calculation Manager** Calculation Manager allows administrators to design, validate, and administer rules in a graphical environment for the core products.
- **Oracle Hyperion Smart View for Office** Smart View provides a common Microsoft Office add-in for Essbase, Financial Management, Planning, and Reporting and Analysis. It can also perform ad-hoc analysis on data from Oracle Business Intelligence Enterprise Edition (BI EE). One of the most underrated features of Smart View is that it is for more than just Excel. Smart View works with Word, PowerPoint, Outlook, and Excel.

You interact with the system primarily through the web. Since there is no local application installed, you can roll the products out to more users. This is just one more way the toolset is able to scale. The primary page users interact with is Oracle Enterprise Performance Management Workspace. The Workspace provides a single place to find and work with the suite of products in a single, multitabbed user interface.

There are newer tools that focus on very specific tasks. For example, the tool Disclosure Management provides an integrated tool for eXtensible Business Reporting Language (XBRL) reporting. Financial Close Manager provides a tool to manage, communicate, and track the entire close process. Financial Close Manager will soon help drive account reconciliation.

The final component to the EPM suite is the financial reporting. There are several tools for reporting, and each one helps serve a purpose. Financial reports work very well for external reporting. Web Analysis, Oracle Business Intelligence Publisher, and Oracle BI EE work best for dashboard reporting.

## Define the Scope

Once you have decided what products you will implement, you need to define the scope of the build. The scope is important to determine what you build and when. There are a couple of steps to defining the scope. The first step is to define the critical success factors, or the main purpose of your project. Since each product does something specifically, you should focus on that goal for the project. For example, if you are preparing for an HFM project, you should be focusing on financial consolidations and external financial reporting. You may find you have the bandwidth for adding some budget collection, or management reporting, but your primary focus should be consolidations. This seems obvious, but many people find themselves trying to sell a project internally or trying to make sure they are maximizing the return on this investment. In HFM, people have all kinds of things you would not think of including in a consolidation system. And a lot of them were good ideas. For example, some things that seem like good ideas are unit information, or inventory details. It might be a good thing to add something like this, but you increase your chances for success by tackling that additional piece in a second phase. Successful attempts at being creative to expand value, more often than not, happen after the main consolidation project is completed.

The second step to defining scope is to focus on the core products. If you have purchased more than one product of the EPM suite, you will need to prioritize your implementation. To focus on what products you will be implementing and when, look at your financial calendar and consider what parts to implement during the year. If the budget is due in six weeks, you may be aggressive, thinking you can get Planning in place first. It is a sizable undertaking trying to implement the full EPM suite all at once. Each product not only has its own design considerations, but often impacts different parts of a company very differently. Consider stepping back, consider what products you will be using both tactically and strategically, and plan how you will put these tools into place.

## 6 Oracle Hyperion Financial Management Tips & Techniques

Finally, you need to define your time line. The time you have to implement should help you prioritize what you can accomplish during the implementation. So, if your time line allows, and you have the appetite for having a large project, you could implement the full suite all at once. I would not recommend it. Many people will think that because you expect some economies of having several projects of the same products set, same users, you can save on things like project management. But the risk of having so many tasks, teams, and changes all moving at once outweighs the benefits.

### **EPM and EPI**

One common mistake people make is failing to understand what it means to design an HFM application. It does not mean to re-evaluate the entire business process. Reviewing and redefining the business processes around HFM is expensive and time-consuming, and not required in order to have a good build and a successful project.

HFM is an EPM tool. That tool has functionality that drives how certain processes will function. Using them for other purposes or not as intended creates issues. Using any tool for a purpose it was not intended always leads to some shortcoming. Unless the tool is meeting some very short-term need, why set yourself up for frustration?

EPI, or Enterprise Process Improvement, is not about a tool. It is not even technology-centric. This is an open evaluation of what you do, why, and how you can do it better. This can be a process that takes years, built in stages as the company grows.

While it is important to understand the current business process, and consider what that future state might look like, the future process does not impact as many of the design considerations as you might think. A good design will allow for future requirements. For example, if I asked a hundred corporate controllers to explain the impact of International Financial Reporting Standards (IFRS) on their business in 2003, I doubt I would have gotten one certain answer. Still, applications that were designed well even before that time will be able to accommodate IFRS.

Consider the effort of evaluating and reviewing every process that impacts the close. If you were going to do that, you could also re-evaluate the reports used and the metrics used to measure the business, and consider reorganizing the business. If you use that logic, the scope can easily expand



beyond what is really needed to complete a successful project. The building of an HFM application is best handled as a discrete step of a business process evaluation and change effort. Remember, this tool is built to do financial consolidations. And while many business process changes may be things worth doing, they are not required in order to create an HFM application. That is true even for an application that will be used to do all of those transformative goals you may have. A good requirements and design meeting should only discuss those items, so the design team can confirm that the design considers the impact of those changes.

## Identify Your Team

Once you have decided on a product implementation strategy, and considered the scope, you need to assemble a team to help implement HFM. There are some key roles you need to fill if you want to be successful. I would group them as internal and external.

### The Internal Team

There are four internal roles: Subject Matter Experts (SMEs), an Application Administrator, Infrastructure Support, and a Project Manager. Having these three roles defined internally doesn't guarantee success, but in every implementation that had issues, these roles were not defined. It is also important to define these roles because they will be required to support the application after you go live.

First, a good team will consist of people who understand the business process and are very familiar with the politics and procedures of the business. You will need people who understand the accounting required, including tax, treasury, management reporting, and especially consolidation accounting. These people are Subject Matter Experts. These SMEs should have some minimum time commitment to the project. These people are important to the project because they will help with data reconciliation, training, and support of the application after go-live. When trying to staff the project, this is one role that people often look to staff by hiring temporary resources. I can tell you, no temporary person you hire will know your business and accounting better than the people in your organization. So your project will suffer if you try to hire temporary resources for this role.

## 8 Oracle Hyperion Financial Management Tips & Techniques

And in the unlikely chance you are successful, when the project is over, the people who learned this new system will leave and take this knowledge with them. So, you should look within your organization for these people.

Second, you need to identify an application administrator. This selection is extremely critical to the success of the project. Many projects have overcome seemingly insurmountable obstacles because of strong, bright administrators, and just as many have struggled to get footing within a company because of distracted or overworked administrators. The best people to select as administrators are people with a finance background. Although HFM uses Visual Basic in spots and has scripting, HFM is not a tool used by the infrastructure groups. It is a finance tool. It is much easier to teach a finance person the technical components of HFM than it is to teach an IT programmer the workings of debits and credits. A person with moderate skills in writing a macro in Excel will have all the scripting skills required to administer HFM. This person will own the bulk of the maintenance after the project goes live. They will be critical in maintaining and supporting the application after you have gone live. This role could be broken out by product or by database. The administrators are usually also responsible for data integration as well. It would be wise to plan for having a backup. I would expect this person to spend between one and two weeks per month supporting the application after the implementation is complete.

The administrator should work closely with the IT support team. While the role is not as dedicated as the other roles, the tasks and timing of the infrastructure tasks are very important. The Oracle EPM toolset is complex and has a steep learning curve.

Finally, every successful project has a strong project manager. It is the role of the project manager to communicate risks, track tasks to the budget, and control scope. For some reason, people seem to discount the value this role has in a project. A strong experienced person in this role brings experience about this toolset to the process. Implementing HFM, or for that matter any of the Hyperion tools, is not like working with any other IT project. It is not an enterprise resource planning (ERP) tool, or a custom tool. The Hyperion tools often cross over departments and roles and are wide-reaching. The process for implementing HFM needs to reflect that. You should not make the mistake of underestimating the value of good communication, thoughtful planning, and strong project management. To be successful, the thing to do first is have a plan, and then make sure you

are communicating effectively and identifying the impact of the process. These are all things that good project leadership and project management bring to your project.

## Your Implementation Partner

The next step is to identify an implementation partner. This part of your team will be easier to find. There are companies who will buy HFM that are lucky enough to have found a person who has consulting experience, or at a minimum has been on several projects and can bring that experience to the team. If your company is not one of those companies, I would suggest having an implementation partner that will give you the best chance of completing the project on time and as close to your intended budget as possible. Besides experience, a strong consulting partner has a network of resources, either internally or across the HFM community, that they can reference to get answers to problems when they come up. I would suggest you look for some key factors when selecting your partner. First, are they certified in HFM? There are several Oracle certifications for different products; it does not help to know Essbase for an HFM project. Second, how many projects has the lead resource worked on? Finally, call their references. You will get some great feedback, hear about pitfalls you won't hear about in a sales cycle, and start to build your own support network for after the project go-live.

Most companies that use HFM are publicly traded, and as such are subject to stringent regulations and reporting deadlines. Look for a partner that can provide a system architecture design that can withstand a single failure of any of the individual components ("redundant services"), or failure of an entire service center (disaster recovery planning).

Do not overlook the value Oracle brings to every project. You need to learn to navigate Oracle support and become familiar with how to get issues logged, tracked, and supported. Not only will this be your lifeline after the project goes live, but it is part of the maintenance you pay for with the product, and you should take advantage of it.

## Support from the Infrastructure Group

How much infrastructure support people need to plan for is a very common concern. Especially if you are migrating to HFM from a consolidation and reporting tool like Hyperion Enterprise or Excel, where IT support was

## 10 Oracle Hyperion Financial Management Tips & Techniques

minimal or none, you will not be sure what type and commitment of resources you need to plan for. During the project, the times when it is critical to have IT support are: installation, security, testing, training support, and go-live. Now each project is different, but each of those phases typically does not require more than a week or two. If you decide to take on more complex security like Secure Sockets Layer (SSL) at every layer, and use a tool like Hewlett-Packard Load Runner to simulate load on the servers, the times can go much higher. Once you are live, the time needed to support HFM drops off significantly for most all clients. If you are live with HFM and the application is average in size and you have a full-time IT resource supporting your HFM application, then something is very wrong. Most people plan for support to add users, support the change control process, and help with connectivity issues. Many routine tasks can and should be automated. A strong implementation partner will help you set those up.

### Create a Project Plan

Once you decide which products you will be implementing, and have pulled together your team, you need to make a plan. Most full implementations of HFM take between five and eight months. During the first month, you establish your requirements and design. The installation, administrator training, and a change control process meeting should also happen during the first month. The second and third month will be focused on build activities. If you are not certain about your design, you can plan for a conference room demonstration or proof of concept. During this time the team will develop rules, create structures (often called metadata), and build reports.

Data should be loaded into HFM as soon as possible. It is important to load data early for two reasons. First, a representative set of data will help the development team determine performance and identify issues during the build. Second, it will help with determining performance issues early in the project.

Once the build has begun, this phase of a project is heavy with data reconciliation. Data reconciliation is very time-consuming, and is the one part of any implantation that cannot be underestimated. The reason this task is so difficult to manage is that, for many users, it is the first time they are working with the software, looking for data, and getting their head around

how HFM will calculate the elimination, allocations, and other numbers. This may be the first time the team or users will see data using the 12 dimensions. That learning curve for working with multiple dimensions can take people time to overcome. The second reason data reconciliation can take longer than you would expect is that it is frankly not glamorous work. It can be very tedious. Some people find themselves easily distracted doing data reconciliation. That slows down the work quite a bit.

Once data has been loaded and is being reconciled, then technical testing can start. It is true that data reconciliation is by its nature a test. But you will need to do some other testing to make sure your system is performing well. You need to load data to ensure that you have valid tests. The data does not have to be final or complete, but it should be representative of what the actual data will be once it's complete. You should plan to test when it will not impact data reconciliation, or when data reconciliation will not skew testing results. The length of time required for testing can vary. There are several types of testing:

- **Unit testing** This type of testing is usually not documented, but is required to ensure that each build step is being completed.
- **Integration (functional) testing** This test is a set of common tasks to ensure that each component when brought together works in one application.
- **Performance testing** This testing is at a minimum a baseline of basic tasks on the production server. At best, this test is a full complement of tests simulating activity from remote sites using a tool like Load Runner.
- **User Acceptance Testing (UAT)** This test is either performed by a subset of power users or as a review of training.
- **Connectivity testing** This test is the final test, ensuring that users can access the system by opening the application site in a browser, and security is in place.

After the application has completed testing, the rollout begins. The rollout is the process for getting users to adopt the system. And to have the users ready to take this ownership, you need to train them. This training will

## 12 Oracle Hyperion Financial Management Tips & Techniques

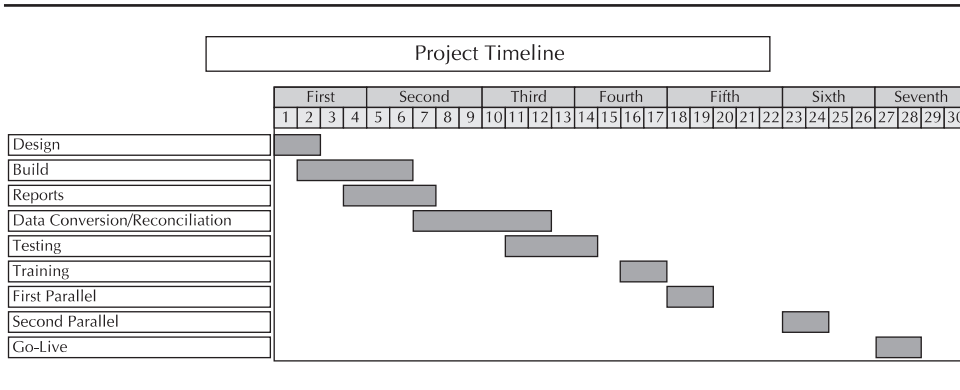
happen before UAT or connectivity testing. Training is often completed for administrators early in the project, close to the design meeting. So the best practice is for end users to be trained as close to the UAT as possible. It is best to have users touch the software as close to the training as possible to reinforce what was learned. This makes the UAT much more efficient. Users have an expectation based on training performance, and can use that to make sure the system is working as expected.

There are many kinds of training, enough that one might fill an entire book on that subject alone. The first consideration you need to make is the groups of people who need training. There is the small group of your core team, and these people need to go to administrator training. This needs to happen either before or after the design. If your project has these people intimately involved, they will be trained during the whole implementation. If not, you will need a deep and thorough training for administrators and a developed plan to transfer ownership.

The second type of training is for your end users. There are couple of options here. The most common are

- **Classroom** Have your users come in for a set class. The biggest benefit is that you control the environment. This works great—people are not distracted or getting pulled from training. This option also can cost the most, especially if people are really spread out geographically.
- **The Road Show** Visit each site with a set training program. Creating multiple teams to visit each group helps defray the costs.
- **Train the Trainer** Have a set of super-users train your team. The benefit here is that training knowledge stays in house and the trainers can transition into a support role for your team after the project is live.

The next step is a simulation of the close using HFM. It is called a *parallel*. I would never recommend fewer than two parallels. Users will need that much time to ensure that they can confidently repeat the close in a reliable way. I would also never recommend more than four parallels. Having too many of these simulations creates quite a bit of fatigue within



**FIGURE 1-3.** A sample HFM project timeline

the user community. Remember, they have to close the books in two or more systems when these final steps are being completed. It can create quite a bit more work than people realize. You want to have at least one of these parallels happen when you are doing a quarter close and preferably a year-end close, if your process changes at all during these times. Since most people do have processes that are done only during quarter close time, they will include one quarter. You should too. You should also choose a month where the work load is not as high to stop using the old system, and just use this new tool going forward. This is called “go-live.”

Although the build is less than a third of the entire build time, looking at Figure 1-3, it is easy to see that the build is the busiest time. You have more going on at that time than during the other project phases.

## Dimension Overview

The purpose of this chapter is to discuss topics you would cover during your design meeting. This section should be used to complement the Administrator Guide, not replace it.

You must understand what a dimension is and how they work in order to move forward with a design. Dimensions are the objects we use in HFM to define coordinates within the database. These dimensions define where data

## 14 Oracle Hyperion Financial Management Tips & Techniques

is stored within the application. Assume that I just had a simple spreadsheet, with columns and rows. When I ask what the value in cell B-9 is, you could determine that B referred to the column and 9 must refer to row. From that you could determine the value in cell B-9. B-9 are the dimensions that tell you where to find the data.

HFM uses dimensions, except there are 12 dimensions for every HFM application. This is more difficult to conceptualize than the 2 dimensions in a spreadsheet. Every application uses 12 dimensions. If you did not use one of the dimensions in your application, you would have to use a default member called [None]. You would need to reference all 12 dimensions to find data in the database, which includes writing a report, entering data, or writing a rule.

To envision the impact of more dimensions, let's take it up a notch; now picture tabs for the spreadsheet. Now picture multiple spreadsheets all grouped in a folder on your computer. To find a number you need to look at one workbook, tab 3, column C, and row 13. Finding your data in HFM isn't much different than that.

It is important to remember that you must consider all 12 dimensions when designing your application. Many people initially think they will be making the application much easier to use when setting the dimensions up by not using one dimension and trying to make two different dimensions fit into one. It is best to think about what are the slices of data you need to define for your reports, then align them with dimensions and logically lay out how they should align within HFM. If you need to use all 12, then by all means, do not feel restricted.

And scope is still important here. It is important to not try to build for every eventuality. You can't plan for every possible change in reporting, so just focus on the reporting and goals you defined at the beginning of the project.

The 12 dimensions used in HFM are

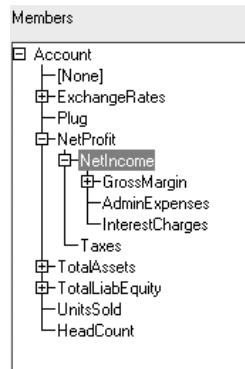
- **View** The View dimension provides calendar intelligence; for example, Month-to-Date, Year-to-Date, and Quarter-to-Date frequencies. This dimension was formerly referred to as Frequency.
- **Year** The number of years you will use for HFM. The Year dimension provides multiple years of a fiscal or calendar year.
- **Period** The Period dimension provides time periods, such as quarters and months. For example, if you need months, there would be 12 periods of data available for a given scenario in a year.



- **Scenario** The Scenario dimension represents a set of data, such as Budget, Actual, or Forecast. For example, the Actual scenario can contain data from a general ledger, reflecting past and current business operations.
- **Entity** The Entity dimension provides a means to build structures for the consolidation. These can be divisions, subsidiaries, plants, regions, countries, legal entities, business units, departments, or any organizational unit.
- **Value** The Value dimension allows for consolidation, elimination, and adjustments and supports translation, which can include the input currency and parent currency.
- **Account** The Account dimension allows you to build a hierarchy of assets, liabilities, revenue, expense, and so on. Each type of account has a specific behavior.
- **Intercompany Partner (ICP)** The Intercompany dimension represents all intercompany balances that exist for an account. This dimension is built dynamically from the Entity dimension.
- **Custom1, Custom2, Custom3, Custom4** The Custom dimensions provide the ability to store other views of the data, such as products, markets, channels, cash flow changes, balance sheet movement, source of data, or types of elimination.

You will also have to define application settings and aliases to build an application; however, they are not dimensions. All of these dimensions are often referred to as the *metadata*. Metadata is a term used to describe data values that define other data. Technically speaking, the dimensions are all data elements that define data within the HFM application. People often use these terms interchangeably when discussing HFM structures.

Now that you know the dimensions, let's understand how they are built within the application. All of the dimensions are built such that each has its own hierarchy. The hierarchy is a "tree" structure where each member has a relationship with other members. The lowest level of this tree structure is called "leaves," like the leaves on a tree. They are also called "base" members, if you want to think of them as the bottom bricks of a pyramid.



---

**FIGURE 1-4.** *A dimension hierarchy*

The tree in Figure 1-4 shows an example of an account hierarchy. The members have “relative” relationships. Some of the obvious ones are “parent” and “child.” As in the example shown in Figure 1-4, the child (Net Income) is below the parent (Net Profit.) The child is below a parent. A parent member can have several children. Two children with the same parent are called siblings. Another relationship used is descendants; those are all members below some parent, including children of children. These relationships are one way the tool creates dynamic lists and allows for drilling from one member to another.

Data is entered into base-level members of the dimensions, not into parent members, with very few exceptions. For example, data isn’t actually stored for parent account members in HFM. The values for parent accounts are aggregated from their children when the data is brought into memory.

### **Naming Convention**

So you are ready to start adding members to your application metadata. You should give some thought as to what to call them, as it could really save you some time and headaches later. When choosing a naming convention for metadata, I actually have a strong preference about this topic that I bring to every design session. Identify the most common or largest source of data for your chart of accounts, and use that for all base

members. This does several things. It simplifies the mapping because you can use the “=” which, simply put, means everything in that dimension maps to everything with the same name. That is only one line of mapping, meaning there is less work that Financial Data Quality Management (FDM) or Oracle Data Integrator (ODI) needs to do to map data values. Even if it isn’t your only line of mapping, your mapping work will be much less. Second, this makes it much easier on you and your end users to reconcile. It is much simpler to see an account A7000, and know that it is the same account and means the same thing in HFM as it does in the ledger. Third, a commonly used structure like what is used in the main ledger is typically the largest source of data and is something your users are already familiar with. This familiarity will speed the adoption of HFM within the company. The structures and dimensionality will be something familiar to the end users.

I would also recommend strongly following the Administrator Guide’s supported characters. Even when an unsupported character seems to work now, it does not mean that it will still work after an upgrade or patch. Special characters like ampersand (&) can cause problems that are tough to identify early on in a project build. That is why they are unsupported.

Finally, while this will be tough to do if you have never implemented HFM before, you should think about what names will help you while writing rules. People often use prefixes like “CF” for cash flow accounts to help them identify which accounts would be used for the cash flow in a list of all accounts. And while I am on this topic, never use a space in any label. It will make troubleshooting a misspelled label in a rule almost impossible. You can’t see the difference in a text file between “Net\_Income” and “Net\_ Income”. (The second one has a space right at the end, and you would never see that in a file.)

You should be consistent when labeling. This will help people follow the system and will speed adoption. Be careful about making the labels too long, though. Remember, you will have to write out some of these names, and the longer they are, the more difficult that will be.

Underscores are supported, and should be used if they help readability. I have found that proper case will do that more often than not, but it is nice to have the option.

## Application Profile

When you are ready to create your application and build hierarchies, you need to create a shell for the members to load into. If you are building these members in a Classic application, you will be using the desktop application. If you plan to build these using EPMA, then you should identify only common members to use from the master library, and build other parts separately.

When first defining your application, you need to define some parameters that cannot be changed unless you rebuild the application. You should not plan to rebuild the application for some time, so you should consider something that you can live with. That does not mean you need to plan to live with this forever. There are times when a rebuild makes sense. This book will cover those cases in the support section, but just know these are not parameters you will be updating and changing without some effort. They are Aliases (Formerly Languages), Calendars, View, and Period.

### Aliases (Formerly Languages)

Let's discuss aliases first. An *alias* is a set of descriptions, typically by language. These can also be as subtle as the difference between tax and corporate reporting. You can specify up to ten aliases for all of the descriptions within HFM. One of your aliases needs to be English. You can't have an HFM application without English. This should not be confused with localization. *Localization* refers to the language used in the menus and toolbars within the application. The Languages refer directly to the descriptions you would modify for all of your metadata and dimensions. Figure 1-5 shows how you might set that up.

Languages don't have to be an actual language, though. I have worked with applications that have used tax labels for state insurance filing as a language in HFM. They could then report from the same chart of accounts both Securities and Exchange Commission (SEC) and state statutory reports with descriptions.

These languages should not be confused with localization. This is the ability to have the menus in Workspace, HFM, and Smart View appear in different languages. These are determined by demand at Oracle. The localization team fills in strings for each menu item with the appropriate

---

Specify the languages your applications will support.

	Language
1	English
2	Spanish
3	French
4	
5	
6	
7	
8	
9	
10	

---

**FIGURE 1-5.** *Adding languages*

translated term for each of the required languages (Spanish, French, Italian, German, Russian, and so on). These strings are made part of the code so that even the release after a localized release, which should only be US English, still has those strings in the code. The strings are surfaced automatically in the browser—there is a code detection that reads the browser’s default language and displays the strings if they are available; otherwise, they display US English. During localization, the product documentation and help guides are also translated.

Experiment with this yourself by adding German, Spanish, or French into your Internet Explorer browser. Go to Internet Options; General; Languages (at the bottom) and add a language (let’s say French). Move the new language to the first position and it will become the default. Restart your browser, and then connect to Workspace; you will see the menus in the chosen language if it is one that Hyperion has localized into. For example, French would appear in menus that normally are in English.

## Calendars

Next you must define the fiscal calendar. You are given three choices for these to get started: Standard, Custom, and Manually Defined. You can modify any of these when you see them, so I have always found that using the Standard calendar gives a great starting point. You can then modify the

## 20 Oracle Hyperion Financial Management Tips & Techniques

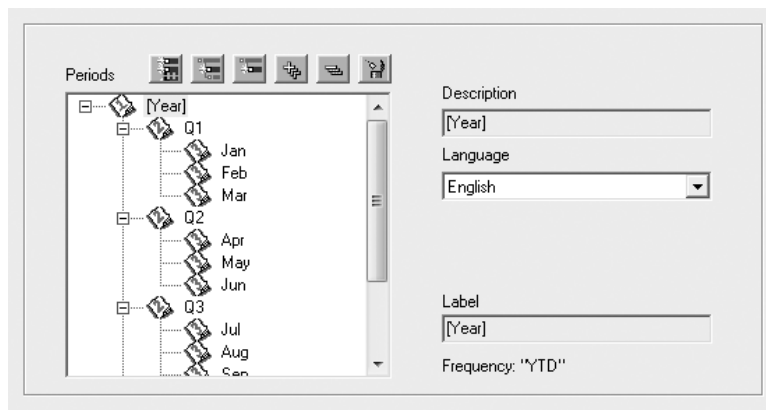
names and descriptions of the months here. After you select the type of calendar and update the time periods for the application profile, default frequencies are created for the application profile. You should not have weekly periods, and definitely not daily periods.

For example, if you select Standard calendar and include quarters and months as time periods, you will see yearly, quarterly, and monthly frequencies. Figure 1-6 is an example of a commonly used calendar. Don't get confused by seeing the "Year" as the top member. This represents the sum of all 12 months, not the Year dimension.

### View

The View is the first to define as part of the application profile. The View dimension provides calendar intelligence. That means HFM will provide values for the view selected. The most common are Periodic, Year-to-Date, and Quarter-to-Date frequencies. If you set the view to Periodic, the values for each month change are displayed. If you set the view to Year-to-Date, the cumulative values for the year are displayed.

You will have to choose both a label and description for the View members you create. As with many of the dimensions, I would recommend something that could be used in your reports for each. For example, I always use the common abbreviation (MTD, QTD, and YTD) for the label, and the longer description (Year-to-Date, Periodic) for the descriptions. Both are helpful and could be shown in reports.



**FIGURE 1-6.** *A sample calendar*

For each frequency, you can enter a label and description for each language that you previously defined. When editing frequencies, you cannot change the label of the YTD frequency, but you can update the description. It is possible to use more relevant descriptions if they make sense for your business, like “Spring” and “Fall” for a retail company. You should consider what you need in your reports to decide this. There are two system-defined frequencies and corresponding views, Scenario View and Periodic. If you had chosen a Manually Defined Calendar as the time period for the calendar, the Frequencies grid would be empty, and you must update the frequencies. You should enter one frequency for each level of the Period dimension, as shown in the example in Figure 1-7.

## Year

The Year dimension seems to spark the most discussion. It is pretty simple; you need only a start year and life of the application. So, it should be a short conversation. Still people struggle with the reality that these aren’t changing. At just about every project, this is the point of the discussion when everyone turns to the most junior person and tries to figure out when they will be leaving the company or retiring, and then they want to add one day to that. Figure 1-8 shows an example of how you might set this up.

While there really isn’t a performance impact to having more years, there are a couple of things to consider. First, people are going to have to scroll through whatever you decide. Does it really make sense to have many

---

Language		
English		
Frequencies		
	View	View Description
Frequency 1	YTD	Year-to-Date
Frequency 2	QTD	Quarter-to-Date
Frequency 3	MTD	Month-to-Date
Frequency 4		
Frequency 5		
Frequency 6		

---

FIGURE 1-7. *Frequencies example*

The screenshot shows a configuration window titled "Time Periods" with three main sections:

- Standard Calendar:** Includes four unchecked checkboxes: "Include Half-Years", "Include Quarters", "Include Trimesters", and "Include Months". Below these is a "Start Month" dropdown menu currently set to "January".
- Custom Calendar:** Includes a "Number of Base Periods" text input field with the value "0" and a "Period Label Prefix" text input field.
- Manually Defined Calendar:** This option is selected with a radio button.

Below the "Time Periods" section is a "Years" section with two text input fields: "Start Year" with the value "2002" and "Number of Years" with the value "19".

---

**FIGURE 1-8.** *Example of years*

years that people will not look at just sitting there to scroll through? Second, open places where data can exist have a remarkable way of finding data to fill them. People fill them by mistake, a rule populates them unintentionally, or test data never gets cleared. That takes up space on the servers and increases costs.

A couple of things to consider here; you can't change the descriptions on years—these years are fiscal, not calendar, years, and you can't add years to the application.

My rule of thumb for number of years is to take the current year and ask how much history, and then add 10. So for an application with 5 years of history starting in 2010, the start year would be 2005, and have a life of 16 years. If you don't think that is enough time, consider this: 10 years ago Intel was still selling the Pentium processor, Windows XP was a year from being released, and Google had yet to top \$80 million in revenue (Google earned over \$23 billion in 2009). A lot can change in 10 years.

### Period

The Period dimension is the last dimension that needs to be fixed. It is also very straightforward. You define months and they roll into quarters or half years, and then those roll into years.

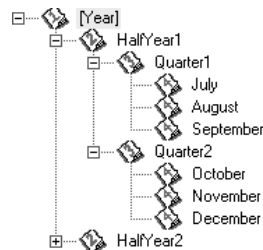


The labeling convention should be like everything else—think of your reports and how you can use both the label and description. I recommend something like in Figure 1-8. You would have a three-letter abbreviation and then the full description.

Finally, you would update the period hierarchy. You will use the time periods and frequencies that you defined. You can make changes to this hierarchy by adding or deleting periods. It really is not a good idea to have a calendar with weekly periods. The performance will be disappointing. I also recommend thinking through whether having a period 13 is really required. Just because it is in your ledger does not mean you need it in HFM. HFM does not need a period 13 to close balances or even to include audit adjustments, so you should question it. Figure 1-9 is an example of how the Period dimension could be set up.

## Currencies

Each entity will be assigned a single default currency. By default, HFM can easily translate any entity into any currency in the application. This can also be done by way of triangulation, which is also supported by HFM. The application will triangulate through the default currency you specified in the Application Settings. The application won't translate into every currency automatically. HFM will only automatically translate when it thinks it needs to, and that means when the currency of an entity is different than the currency of its parent. Other translations will require the users to force the translation.



**FIGURE 1-9.** *Period hierarchy*

## 24 Oracle Hyperion Financial Management Tips & Techniques

While currencies are not a real dimension, when you define currencies, HFM will create members in the Value dimension, Custom1 and Custom2, described later in this chapter. You must provide a label and description for each currency you wish to use. This is another design consideration you should give some careful thought to. You do not want to have to add many currencies later, as doing so has a significant impact on the database. You also do not want to have too many unused currencies, because that can also have a performance impact. I would suggest having all the currencies you do business in now, and those you are certain you will use in the next three years. The labels and descriptions for the currencies should have some logical convention. So consider what your reports require. To that end, it makes sense to use ISO labels and descriptions. As shown in Figure 1-10, use AUD and Australian Dollars, and not something like AUS and Australian \$.

There is no need for currencies with the same type (EUR and EUR1 or EUR\_old). Since these are really separate dimensions, you can enter several EUR to USD rates, even with the same period, all for different entities.

### Application Settings

Once the application has been created, you are ready to start building it. The first step is to define some common application settings. Unlike the profile, these can be changed. It is important, though, to know the impact

---

Members					
	Currency	Description(E	Scale	TranslationD	DisplayInCT
1	AED	United Arab Emi	0	M	<input checked="" type="checkbox"/>
2	AOA	Angolan Kwanz	0	M	<input checked="" type="checkbox"/>
3	ARS	Argentine Peso	0	M	<input checked="" type="checkbox"/>
4	AUD	Australian Dollar	0	M	<input checked="" type="checkbox"/>
5	BDT	Bangladesh Ta	0	M	<input checked="" type="checkbox"/>
6	BGL	Bulgarian Lev	0	M	<input checked="" type="checkbox"/>
7	BHD	Bahraini Dinar	0	M	<input checked="" type="checkbox"/>
8	BND	Brunei Dollar	0	M	<input checked="" type="checkbox"/>
9	BOB	Boliviano	0	M	<input checked="" type="checkbox"/>
10	BRL	Brazilian Real	0	M	<input checked="" type="checkbox"/>
11	BSD	Bahamas Dollar	0	M	<input checked="" type="checkbox"/>

---

FIGURE 1-10. *Sample currencies*

of each setting. You will need to come back to these settings once you have defined the accounts. The following list shows some key application settings.

- **ConsolidationRules** This identifies whether consolidation rules will be customized. Specify one of these values: Y, R, or N. Specify Y to use the custom rules written in the Consolidate() subroutine. R will derive the proportional and elimination value in the Value dimension.  
N will use the default consolidation and eliminations. Whenever possible, use R to optimize consolidation performance if your application can use the default HFM consolidation rules.
- **DefaultCurrency** Identifies the default currency for the application. Most applications will use the reported currency of the parent company.
- **DefaultRateForBalanceAccounts** The currency rate account that contains the translation rate to use for Asset or Liability accounts. This is the end-of-period or end-of-month rate.
- **DefaultRateForFlowAccounts** The currency rate account that contains the translation rate to use for Revenue or Expense accounts. This is the period average rate account.
- **DefaultValueForActive** Identifies the default value for the Active account. This attribute is required even though it is for OrgByPeriod applications only. Specify 0 if the child entity is considered inactive by default and will not consolidate into the parent entity. Specify 1 if the child entity is considered active by default and consolidates into the parent. Since most of your new entities will be consolidated, you should choose 1 unless you are building an OrgByPeriod application. In that case you should choose 0. When you choose 0 for Organization by Period applications, no historical periods are impacted when a new entity is added or an existing entity is added to a new parent. Organization by Period has many other considerations covered later in this book.

## 26 Oracle Hyperion Financial Management Tips & Techniques

- **EnableMetadataSecurityFiltering** Metadata filtering is a feature that allows users to see only the members to which they have access, and not necessarily see the entire structure of the database. You can filter on Scenario, Entity, Intercompany Partner (ICP), Account, Custom1, Custom2, Custom3, or Custom4. The default for this attribute is N, but choose Y when you want to limit people's access.
- **FDMAppName** Name of the Oracle Hyperion Financial Data Quality Management application. This facilitates "drillback" from HFM into FDM.
- **ICPEntitiesAggregationWeight** Identifies the percentage of intercompany partner entity [ICP Entities] amounts that aggregate to the [ICP Top] within the Value dimension. It is almost always 1.
- **MaxCellTextSize** Identifies the maximum number of characters that can be used for cell text. You can enter a positive number up to 2,147,483,646, or 1 for no limit. The default of 8000 will be more than enough for most people.
- **MaxDocAttachmentSize** Identifies the maximum number of bytes for any document attachments. You can enter a positive number up to 2,147,483,646, or -1 for no limit. You should consider limiting this as some people will load excessively large files.
- **MaxNumDocAttachments** Identifies the maximum number of document attachments per user. You can enter a positive number up to 2,147,483,646, or -1 for no limit, which is the default as well.
- **NodeSecurity** Enter Entity to check node data based on security access for the entity and Parent to check node data based on security access for the parent. Most applications have Entity. This setting will control who can access the Node of the Entity. These are the value members with square brackets. You might use Parent when another person will be pushing journals or data down to entities earlier in the close process.
- **OrgByPeriodApplication** Specify Y to use Org by Period or N to use only one organizational structure. If you are planning on building complex ownership structures in the entity dimension, you should plan on using Org by Period.

- **SupportSubmissionPhaseforAccounts** If you are planning to submit part of the chart of accounts at different times for process management, you will need to set this to allow for phased submissions. Your options are Y or N, and the default is N.
- **SupportSubmissionPhaseforCustom1-4** If you are planning to submit some of the custom members at different times for process management, you will need to set this to allow for phased submissions. Your options are Y or N, and the default is N.
- **SupportSubmissionPhaseforICP** If you are planning to submit different ICP members at different times for process management, you will need to set this to allow for phased submissions. Your options are Y or N, and the default is N.
- **UsePVAForBalanceAccounts** PVA is a method of translation where the periodic value is translated for each month, and the months are summed to give a year-to-date balance. If you select Y, this translation method would be used on Balance accounts. Balance accounts typically use the VAL method, which is translation at a point in time.
- **UsePVAForFlowAccounts** PVA is a method of translation where the periodic value is translated for each month, and the months are summed to give a year-to-date balance. If you select Y, this translation method would be used on Flow accounts.
- **ValidationAccount** This is required for process management. The validation account must equal zero before a process unit can be promoted or rejected. The validation account also serves as a locking account, meaning that it must equal zero before you can lock an entity. You can specify a validation account for Submission Phase 1 to 9.

Figure 1-11 shows how we would set these application settings up. Now that you have a profile and defined application settings, you are ready to build the rest of your metadata.

## 28 Oracle Hyperion Financial Management Tips & Techniques

Members								
	DefaultCurrency	DefaultRateForBalanceAccounts	DefaultRateForFlowAccounts	UsePVAForBalanceAccounts	UsePVAForFlowAccounts	ICPEntitiesAggregationWeight	DefaultValueForActive	ValidationAccount
1	USD	EOMRate	AvgRate	<input type="checkbox"/>	<input checked="" type="checkbox"/>	1	1	

Members								
	ConsolidationRules	OrgByPeriodApplication	NodeSecurity	UseSecurityForAccounts	UseSecurityForEntities	UseSecurityForScenarios	UseSecurityForCustom1	UseSecurityForCustom2
1	Y	<input type="checkbox"/>	Entity	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Members								
	UseSecurityForICP	EnableMetadataSecurityFiltering	MaxCellTextSize	MaxDocAttachmentSize	MaxNumDocAttachments	UseSubmissionPhase	SupportSubmissionPhaseForAccounts	SupportSubmissionPhaseForCustom1
1	<input type="checkbox"/>	<input type="checkbox"/>	2048	0	0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Members								
	SupportSubmissionPhaseForCustom2	SupportSubmissionPhaseForCustom3	SupportSubmissionPhaseForCustom4	SupportSubmissionPhaseForICP	ValidationAccount2	ValidationAccount3	ValidationAccount4	ValidationAccount5
1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>				






<input type="checkbox"/> Disable Combo Boxes       					
ValidationAccount6	ValidationAccount7	ValidationAccount8	ValidationAccount9	FdmAppName	

FIGURE 1-11. Application settings example

## Scenario

The Scenario dimension is used to identify the type of data that will be stored for all periods, entities, and accounts. It is a version of a projected course. The most common is “Actual.” That would be all actual data reported. Other types of data include “Budget” and “Forecast.” You could have “Versions” of each. For example you may have Budget1, Budget2, Budget3, and a final budget submission just called Budget. Each of these examples tells a different story of the data in HFM.

You can determine how many you need by asking yourself how many of these versions you review and report on each month. These reports are often called *variance reports*, and they show the differences between these multiple Scenarios.

Like all the other dimensions, the Scenario dimension has a hierarchy, but it is flat. One could have a structure, but it will only serve to help people see the relationship between other Scenarios. You may want to have the “Final Budget” be a parent, and the versions of working budgets are children.

There are some important settings specific to the Scenario dimension that should be considered.

- **ConsolidateYTD** Enter Y for YTD or N for periodic. The decision point here is how should HFM run its consolidation; Org by Period applications will always be periodic.
- **DefaultFreq** This identifies the periods defined in the Application Profile for which data input is valid. For example, Monthly indicates that you can extract input data only in month-based periods.
- **DefaultView** YTD or Periodic. Typically, Budget and Forecast would be periodic and Actual is YTD. This default view impacts rules as its default view.
- **DefFreqForICTrans** Required for Intercompany Transaction Module, this setting identifies the default frequency for intercompany transactions.
- **EnableDataAudit** Y to automatically audit all accounts (even accounts that have EnableDataAudit set to False), or O to audit only accounts with EnableDataAudit set to True. You would choose N to disable auditing for this scenario.

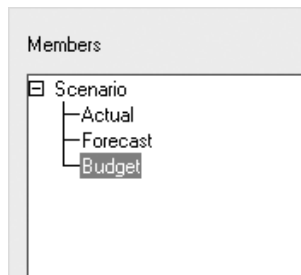
## 30 Oracle Hyperion Financial Management Tips & Techniques

- **MaximumReviewLevel** This attribute is required, but is not used by HFM in any way. You could put any number and it does not impact any functionality. Still, you still need a value from 1 to 10.
- **PhasedSubmissionStartYear** Identifies the start year for phased submissions in process management. This is a new feature of Release 11.1.2.
- **SupportsProcessManagement** Y to enable Process Management, N to disable Process Management, or A to enable Process Management and E-mail alerting.
- **UsesLineItems** Y if the scenario can accept line item detail. This is useful for accounts that require line item detail in Actual, but not in Budget.

Figure 1-12 shows the three most commonly used scenarios and their settings: Actual, Forecast, and Budget.

The next two settings are key settings that need to be considered for setting up scenarios. `ZeroViewForAdj` and `ZeroViewForNonadj` are both required attributes that inform applications how to handle missing data. They may be set to either Periodic or YTD, and HFM derives either Periodic or YTD values for missing data accordingly.

Why do you even need these settings? These settings are there simply because you do not want to load zeros into the database. A zero is actually something in the database. A zero takes up space and requires the system to



---

**FIGURE 1-12.** *Most common scenarios*



handle it, either reading or writing it to the database. Not having zeros in the application will really increase performance.

There is a rare exception when you need to load zeros, and that is when the zero view settings you defined for the scenarios are not valid for all entities; for example, when you have to load periodic data, but the default view is year to date. This would create cases when the periodic change is zero, you need to load that zero so that the year-to-date number is correct. But you really should exhaust your options trying to get the data file in a format that is consistent.

The first difference to note in these settings is that one is for handling Adjustments (journals) and the other is for all other data points.

You should consider how you plan to look at the data; one way is selecting Period view. This is most common for Budget and Forecast scenarios. Let's assume I have selected a Budget scenario that I am viewing periodic. If ZeroViewForAdj or ZeroViewForNonAdj is set to Periodic, HFM displays *a derived value of zero* for missing data, and it is treated as a zero for the current period change. I would see a zero in the database. If I changed my view to Year to Date, HFM displays *a derived value that equals the sum of previous periods' values*. The year-to-date number would suggest there was no change for the month. Looking at Figure 1-13, you can see how HFM populates these numbers.

If you were looking at actual data, you would likely prefer a Year to Date view of the data. If ZeroViewForAdj or ZeroViewForNonAdj is set to YTD, and I was viewing the data Year to Date, HFM displays *a derived zero for the year to date*. That would mean the month change would have to equal the negative of the sum of the prior months, as shown in Figure 1-14.

---

	January	February
YTD	100	100
Periodic	100	0

---

**FIGURE 11-13.** *Period change of zero*

---

	January	February
YTD	100	100
Periodic	100	0

---

**FIGURE 1-14.** *Year-to-date change of zero*

### Variance Reporting

For reports that compare two scenarios, for example Actual to Budget data, you have two options for building this. You can either build two scenarios, or build the second view in the custom dimension. Since you only have four customs in the current version, if you are out of dimensions the decision may be made for you. But the advantage of building this type of reporting into a custom over a scenario is that it is easier to manage copying data and journals.

There are some great features in the reports that open many more doors. There are some great ideas for these reports in Chapter 4.

### Constant Currency

Similar to the issue with variance reporting, you can make managing the application easier by having Constant Currency Scenarios in a custom dimension over having a separate scenario. It can help to use scenarios to look at Actual data at Budget rates, Last Year rates, and other rates to evaluate the currency impact on your financial statements. That would require adding a scenario and copying data and rates from other scenarios. The other option to creating scenarios is using one of the custom dimensions. The driver for making this decision is simply performance and availability of dimensions. If you have used all your custom dimensions, then it is not an option to use them. If they are available, then you could consider that. So if dimensions are available, you only need to consider performance. The performance impact is that you would potentially double the data within your subcube. This can slow down the consolidation. But there is a big advantage to putting this into a custom dimension: You never have to worry about data in one scenario changing and not flowing into the other translation rates you want to see. Older versions of HFM do not have the newer memory-handling features that minimize this issue.

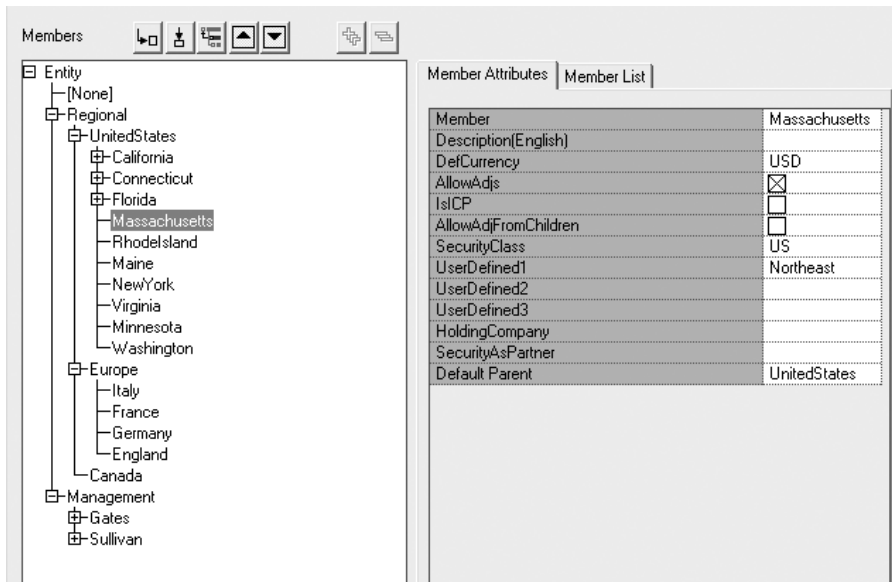
I would suggest using scenarios, as you have more control over when data is populated into the scenario for evaluation; the special scenario can be consolidated as a separate process.

## Entity

The Entity dimension should be used to represent the organizational structure of the business. This is usually done with legal entities, or cost and profit centers at the base level, then consolidating by region, responsibility, or by a defined legal structure. Each entity stores data for the necessary consolidation points. During a consolidation, all the children of a parent are loaded into memory for the system to aggregate all base-level data to store for the parent. This works very well for the structures that are common in consolidation systems. Still, very flat entity dimension structures can experience performance issues due to the amount of information that must be cached. HFM has had many improvements in recent releases that resolve memory issues, and these improvements along with subcube enhancements are discussed later in this chapter. However, versions used before System 9 and applications with large entity structures should consider adding parents, if for nothing else than to improve performance. These additional entities can help make the application more user-friendly, providing more groupings for reporting and drilling on the data.

There are several settings for the Entity dimension to consider when adding new members to the Entity dimension. Figure 1-15 shows a sample Entity. The following list shows the key attributes that must be defined; you should spend extra time understanding the impact of these attributes.

- **Member** Labels are restricted to 80 characters, can be alpha and/or numeric. Don't forget to put some thought into your naming convention.
- **DefCurrency** This is the default currency for the entity. It should be the functional currency of the entity.
- **AllowAdjs** Having this attribute set allows journals to be posted to the value member Entity Currency Adjustment and the Parent Currency Adjustment members. Now with this enabled, it also allows HFM to run



**FIGURE 1-15.** *Sample Entity*

Sub Calculate there. This feature is not well known, but it can speed your consolidations. If you don't have journals there, there is no need to run calculations on these members.

- **IsICP** This attribute determines whether this entity will be an intercompany partner for any intercompany transactions. Flagging this as Y (yes) will create a member in the ICP dimension, and allow for input of intercompany accounts to an ICP member with the same label as the entity. Selecting R (restricted) will allow for the same conditions as Y will; however, it will prevent people from posting to an intercompany partner with the same label as the entity you are entering data in. This prevents people from recording intercompany activity with themselves. I would recommend using R unless you are certain you are not reporting at the lowest legal entity in the application.
- **AllowAdjFromChildren** Having this set allows journals to be posted to the value member Contribution Adjustment and the Parent Adjustment members.

- **UserDefined1–3** User-defined attributes are used to enhance the flexibility of rules and for creating lists. Rules can be conditionally based upon these fields.
- **HoldingCompany** This field is used to identify the Holding Company for this entity. It can be referenced in the rules. When you are building rules in the system for the consolidation, you will see how this field is critical.

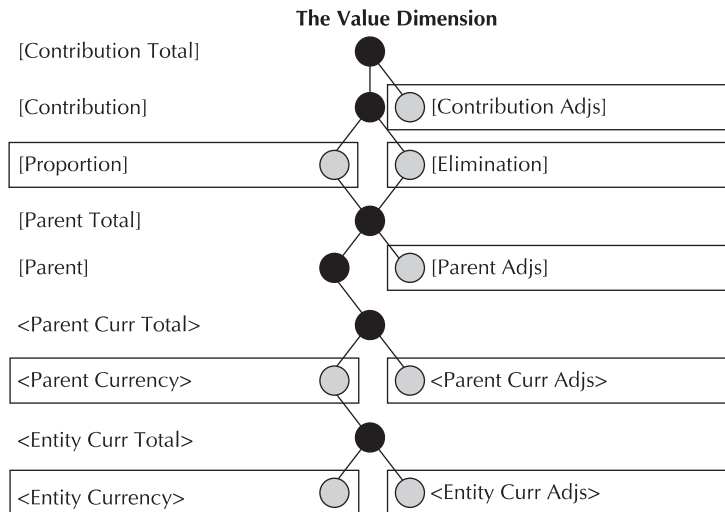
## Value

The most important dimension is the Value dimension. It is the part of the product that drives the consolidation in HFM. When you really understand how the Value dimension works, you understand how HFM really does consolidations. It is critical for writing rules. It isn't just how HFM brings the data together, but it tells you how the rules are working to move data up the entity structure to consolidate the data. The Value dimension is a system-generated, system-maintained dimension within HFM. You can't modify it directly.

I say directly because the Value dimension has members based on currencies you add, and you can make some minor changes to default members.

The first thing you notice when you look at the Value dimension is that all of the currencies you add in the application are replicated there. All the currencies in the Value dimension are referenced by something called a *triplet*. A triplet is a grouping of an input member, an adjustment member, and a total. For example, if you had a USD currency you would see for that currency a USD, USD ADJ, and USD Total. You should see a triplet for every currency you have in the application.

So let's talk through the Value dimension. Figure 1-16 is an example of the Value dimension and how data moves from one member to another. Data is loaded only to <Entity Currency>. Every entity has only one default currency. So even though you see all of the currencies in the Value dimension, you can only load to one—the <Entity Currency>. You can make a local currency adjustment in HFM through the journals module in a member called <Entity Curr Adjs>. These two members sum to a total called <Entity Curr Total>. This is also a triplet, and it points to the currency triplet that is the default currency of the entity. If I had an entity with a functional currency of EUR (the euro),



**FIGURE 1-16.** *The Value dimension*

and I loaded data to Entity Currency, I would see the values I loaded in both the value member EUR and <Entity Currency>. That would be true when I load the data, if I chose EUR or Entity Currency in my load file.

The reason is that the <Entity Currency> and for that matter, all of the value members that relate to the entity, identified by opening and closing angle brackets (“<” and “>”), are just pointers to the currency defined in the Value dimension. They aren’t “real” members that store the data. When you load data to an Entity member of the Value dimension, it is valid anywhere the entity appears in the application.

Once we have the functional currency and adjustments completed, we can do translation. We want to translate the entity before we do any other calculations like eliminations or ownership calculations. By default, HFM will translate to the currency of the parent. If local currency and parent currency are different, then HFM will by default have translation. First, the data in the <Parent Currency> is cleared, then HFM takes <Entity Currency Total>, runs the Sub Translate routine, and moves <Parent Currency>. Once the data moves to <Parent Currency>, the Sub Calculate rules are run again.

<Parent Currency> also has an adjustment member you can add adjustment values to called <Parent Curr Adjs>. Those journal values are made in the parent currency of the entity. Both <Parent Currency> and the <Parent Curr Adjust> are added together on the fly to give <Parent Currency Total>. Once you move to the next member, you will notice that the brackets change from angle brackets (< >) to square brackets ([ ]).

The next member is [Parent]. The Parent member defines the first member that is part of the Node. There is also a [Parent Adj] and [Parent Total]. These members are also in the parent's currency, but unlike <Parent Currency>, they are part of the Node.

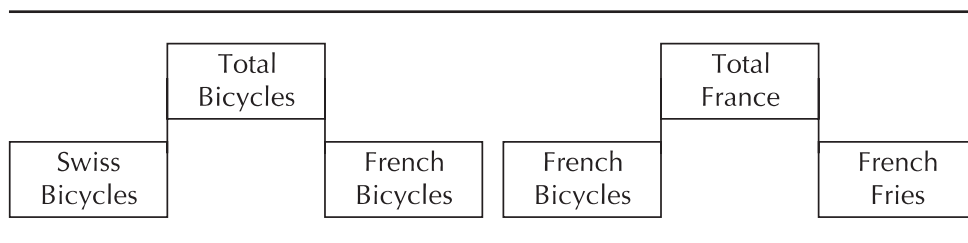
The Node is the unique relationship between a parent entity and each of its children. For example, if I were an entity that made French bicycles, I could roll up to two parents. I could roll up to Total Bicycles and Total France. Figure 1-17 shows this example.

The eliminations and consolidation calculations would likely be very different depending which parent it consolidates. The Value dimension allows this.

The [Parent Total] using consolidation rules, either default or custom, will write values to [Elimination] and [Proportion]. Eliminations occur after any Proportion has been calculated (any non-100-percent consolidations), and the total is stored.

## Account

The Account dimension represents a hierarchy of natural accounts. Typically, the accounts store financial data for entities and scenarios in an application. Each account has a type, such as Income or Expense, that defines its accounting behavior. You can use accounts for other types of data, like headcount, units, or inventory. Those accounts typically use other



**FIGURE 1-17.** *The Node, the relationship of a parent and child entity*

## 38 Oracle Hyperion Financial Management Tips & Techniques

types, like Balance. The account type is one of the first required settings after you have defined the label and description.

- **Asset, Liability** Accounts used on the balance sheet; these do not recognize period changes. If you had an asset Cash that had 10,000, it would be 10,000 year to date and 10,000 periodic.
- **Expense, Revenue** Accounts used for the income statement; these accounts do show period change. Revenue was called Income, but they are the same. If you had an revenue account called Sales that had 10,000, it would be 10,000 year to date and the difference between the months for the periodic.
- **Flow** Behaves like an Expense or Revenue account, but will not translate.
- **Balance** Behaves like an Asset or Liability account, but will not translate.
- **Balancerecurring** Allows you to enter data in one period and have it carry forward until the end of the year. If you entered 500 in January, then 500 would appear for all the months after January. This would only carry until the end of the year, and then the data needs to be entered in the next year.
- **Currencyrate** These accounts allow use of currencies in Custom1 and Custom2. A commonly used account would store the end of month rate (EOMRate). Then you can enter a translation rate. The rule of thumb here is to enter the rates as they translate from Custom1 to Custom2. So if you were going to translate a EUR entity to a USD parent, the data would be A#EOMRate.C1#EUR.C2#USD. These accounts are Balancerecurring, so if you don't enter a rate in the next month, it will use the last one entered.
- **Grouplabel** Used to group accounts, Grouplabel will not store a value. You may want a group called "Administrative"; this would be a nice way to group all the tax and translation rate accounts.
- **Dynamic** Indicates that the account is calculated dynamically, "on the fly." The accounts are typically simple ratios and require a special rule.



Once those account types are defined, you need to consider some of the other settings.

- **CalcAttribute** This field allows you to describe the calculations in the rules file that are done for this account. It can be viewed by users in grids and forms. You can enter in plain English what the rule is doing. This feature is not used enough in HFM applications. A well-designed application will include this.
- **DefaultParent** The default parent for the account.
- **EnableDataAudit** Y to enable account auditing or N to disable auditing. To use this functionality, you will need to also turn this on using the Scenario attributes.
- **ICPTopMember** The ICP top member for the account. If you want to prevent entry or usage of [ICP None], this setting can force users to identify a specific intercompany partner when they use this account.
- **IsICP** Identifies the account as an intercompany account. Y will make the account valid for all members of the ICP dimension, N will allow ICP None to be the only valid member for loading data, and R will make the account valid for all members of the ICP dimension except for the ICP member that has the same name as the entity loading data.
- **PlugAcct** Consider this a suspense account for identifying and resolving intercompany matching issues. It is required for accounts to self-eliminate if the consolidation rules are not used.
- **IsCalculated** This should be used on all base calculated accounts; it prevents input and tells HFM to clear data values before rules are run at key points of the consolidation.
- **IsConsolidated** Allows the account to consolidate in the Entity/Value dimensions. If this is not selected, an account will not move past <Parent Curr Total> in the Value dimension.
- **Submission Group** For Phased Submission, this setting will identify the submission group.

- **UsesLineItems** Y if the account uses line item detail and N if the account does not. This also needs to be flagged in the scenario if you want to have line item detail.
- **XBRL Tags** This is a field where you can enter XBRL tags for the account. This setting is not related to Disclosure Management (DM). I would not use it for any substantive XBRL reporting.

Figure 1-18 shows a typical account and the key settings described.

### System Accounts

HFM provides a set of accounts that can be used to help with complex consolidation and calculations. These accounts help drive consolidation rules, and that is covered later in the book when Rules are covered in Chapter 3. All system accounts are Balance accounts except for the Active

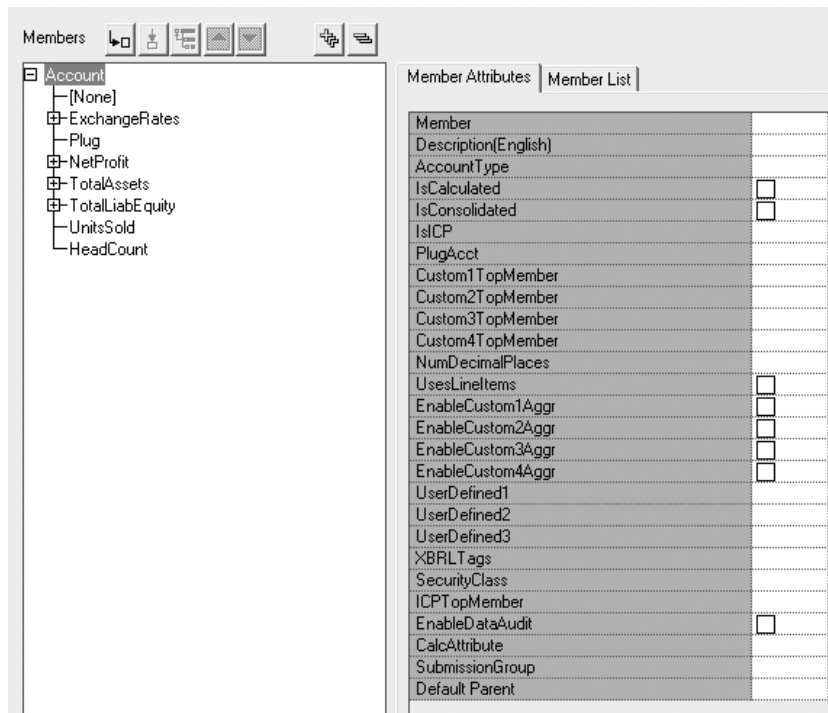


FIGURE 1-18. The Account dimension

account, which is a Balancerecurring account. You can modify the description, security class, and the decimal location for system accounts.

- **[PCON]** Percent consolidation
- **[POWN]** Percent ownership
- **[DOWN]** Percent of direct ownership
- **[PCTRL]** Percent control
- **Active** Determines whether the entity should consolidate in the period
- **SharesOwned** Total number of shares owned
- **VotingOwned** Number of voting shares owned
- **SharesOutstanding** Total number or percentage of shares outstanding
- **VotingOutstanding** Number of voting shares outstanding
- **Shares%Owned** Calculated based on above account information
- **Voting%Owned** Calculated based on above account information

## Intercompany Partner (ICP)

The Intercompany Partner dimension provides detail for all intercompany balances that can exist for an account. Oracle Hyperion Financial Management can track and eliminate intercompany transaction details across entities and accounts. This dimension is defined when you flag an Entity as an intercompany partner. The member labels will mirror those entities flagged.

Figure 1-19 shows the screen where you will set up an application for intercompany transactions. You must perform these actions:

1. Indicate the accounts that perform intercompany transactions and indicate a plug account for each intercompany account (IsICP and PlugAcct attributes in account metadata).
2. Indicate the entities that perform intercompany transactions (IsICP attribute in entity metadata).

## 42 Oracle Hyperion Financial Management Tips & Techniques

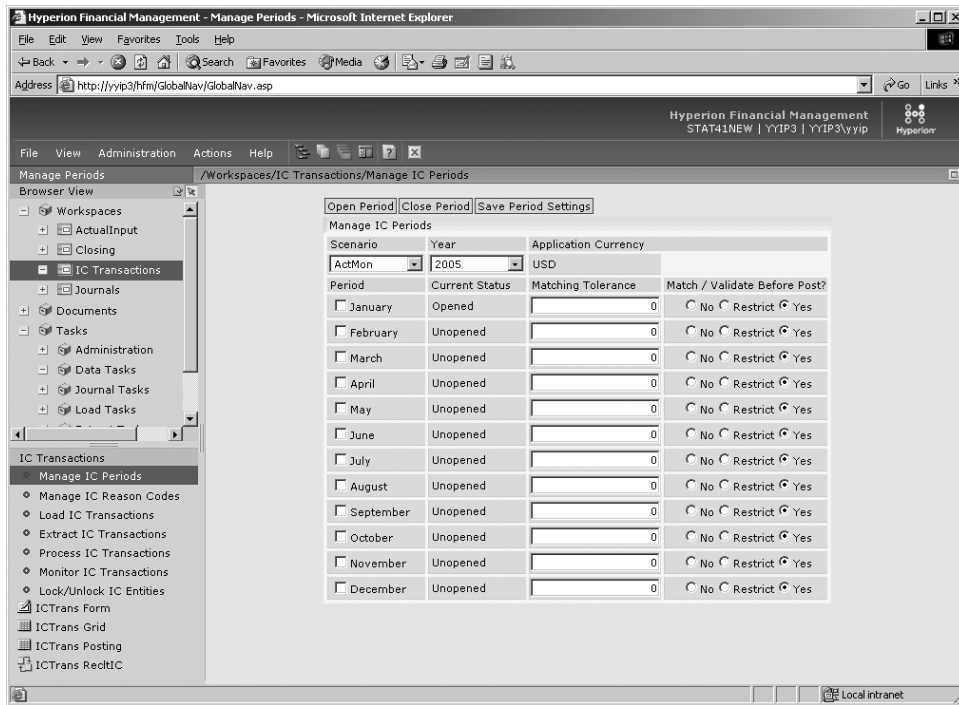


FIGURE 1-19. *Configuring Intercompany*

The plug account created should always be intercompany. It is valuable to see the matching by intercompany partner when trying to resolve a data issue.

## Custom Dimensions

In addition to the eight system-defined dimensions, Hyperion Financial Management provides four custom dimensions. You use the custom dimensions to store additional detail, such as products, measurement adjustments, or balance sheet movement. For example, you can have a custom dimension for products associated with your Sales account that you can use to track sales by product.

It is best to think of custom dimensions as another view of the accounts. For example, you may want to view your operating Expense by Function or Department. You may want to see Revenue by Product. In those cases you can put Function and Product in a Custom dimension. Then, by attaching the custom dimensions to the accounts, you create a valid intersection of the account and Custom dimensions. Now in this example, you do not have any overlap of the customs. Revenue is only valid for Product and Expense is valid only for Function. Since there is no overlap, you can put both members in the same Custom member.

To attach them to the accounts, you only need to do two steps. First, attach the top Product member to the Revenue accounts, and the top Function member to the correct Expense accounts. Second, enable Aggregation for each of the accounts.

## Consolidation Methods

Consolidation methods are used during the consolidation process to allow HFM to properly calculate the amount that is written to the [Proportion] Node. Normally, there are three consolidation methods you might use depending on the strength of the parent company's control or influence: full consolidation, proportionate, and the equity. The basic principle consists of replacing the historical cost of the parent's investment in the company being consolidated with its assets, liabilities, and equity. You can actually create any names you want, but those are the most common. The methods you define will automatically generate in the [ConsolMethod] system list for the Custom1 dimension. The Consolidation method can be populated either manually through data load or data entry, or populated in the rules, which are based on the ultimate percent control for the entity. Consolidation methods are also used to write to the [Elimination] member.

POWNMIN is a keyword you use for the method corresponding to the Equity method. The settings in this table are used by the Ownership Calculation routine to compute the percentages of control, the ultimate percentages of ownership, and to assign the percentages of consolidation and the consolidation methods for legal consolidation. Using POWNMIN, the percentage of consolidation that is assigned for the Equity Company corresponds to the percentage used in a staged consolidation POWNMIN calculation.

## 44 Oracle Hyperion Financial Management Tips & Techniques

These methods are ways you can consolidate the data in your application. You can build them in whatever way you need to run the consolidation. I have seen some very common ones. I am sure each reader will have seen some twist on each of these, so I will keep the examples generic.

Full consolidation is used when there is majority ownership or control (greater than 50 percent of voting shares) by one parent of its subsidiaries. The two company's financial statements are combined by account, with any adjustments and eliminations. This can mean the consolidated accounts have only part of the values consolidated to a given parent. For example, if you have 100 in revenue, but own 65 percent, you would only see 65 consolidate to the parent. Statement of Financial Accounting Standards (SFAS) 94, issued in 1987, states that majority-owned subsidiaries (more than 50 percent of the voting stock has been acquired), should be fully consolidated and that accounting by the Equity method (generally used for affiliates less than 50 percent owned) is not a substitute for information provided by fully consolidated financial statements.

The Equity method is used when there is minority ownership or control with significant influence (between 20 percent and 50 percent) by one Parent of its affiliates. The Parent entity will reflect ownership of equity with entries to specific accounts, as opposed to all accounts. The equity method is allowable for affiliates less than 50 percent owned as opposed to subsidiaries more than 50 percent owned.

The Cost method is generally used when there is minority ownership or control without significant influence (less than 20 percent of the voting shares) by one Parent of its affiliates. This method is similar to the Equity method, with the exception that entries are put into the system only when dividends are paid.

The Proportion method is used for joint ventures, a method of including items of income, expense, assets, and liabilities multiplied by a firm's percentage of participation in the venture. There is some question of the future of this method with expansion of IFRS.

Other methods such as Joint Venture (JV) or Associate can be used and modified to help with issues you may have. You can create what you need.

Once you have determined your methods and built them into your application, they need to be assigned to the entities. There are a couple of ways

to assign the consolidation method to an entity for use during consolidation. The method can be assigned through the ownership console, manually through data load or data entry. The method can also be assigned by the Calculate Ownership routine.

## Best Practices for Design Dimensionality

One of the most important design considerations you can make with HFM is thinking about the subcube. In recent releases the product has gotten much better at handling the memory limitations the subcube presents. In releases prior to 4.1, Hyperion strongly recommended a limit of 100,000 records for any one subcube, based on a 12-month Application Profile. This limit was not a hard and fast limit; one could see subcubes larger than 100,000 records completing consolidation without consistently failing or giving an error message. This made it very difficult to identify and resolve these issues. The large subcubes actually bring you closer to the real issue, which is the Maximum Number of Data Records in RAM (MaxNumDataRecordsInRAM) to the 1,500,000 threshold. There is a limit of data records you can load into memory. Each record is about 200 bytes for a 12-month application. That provides the recommended 300 MB for data caching. You can modify this limit by changing the MaxNumDataRecordsInRAM setting in the registry. But this should only be changed if you are sure of the results.

This was a limitation of the memory available in Windows for the service. So the larger the subcubes, or the larger the number of subcubes created, the greater the risk of running into this record limit of 1,500,000. Oracle added some significant improvements, to manage the subcubes, such as 64-bit version of HFM, Lazy Copy, Paging and some options to manage the subcubes in the registry. Since that significant improvement in memory management in HFM, rules issues have become the most common performance issue, more than even bad subcube design. However, large or poorly planned subcubes can still negatively impact performance. The rest of this chapter focuses on the key considerations you must take into account to have a good design.

## The Subcube and Its Impact on Your Design

So now that you understand the subcubes are important, let's define what makes up a subcube so you can better design them. The account, ICP, and customs for a period make up the subcube. Data for a different currency, year, or scenario exists in separate subcubes.

Each subcube is defined by the Page dimensions and contains all the members of the Subcube dimensions. The Page dimensions are Scenario, Year, Entity, Value and View. The Subcube dimensions are Account, ICP, Period, Custom1-4. The subcube consists of the stored data records for these combinations of dimensional intersections. That is, it is the actual number of populated records that make up the size of your cubes. You can estimate the size of a subcube by multiplying the number of members in the Account and Custom dimensions to determine all possible intersections. You can determine the largest subcube by finding your most dense entity, usually at the top of your Entity dimension. Data density can have a big impact on the size of the subcubes. For example, you could have 1,000 accounts, and 100 of each custom valid for each account. That is not something anyone would call a large application, but  $(1,000 \times 100 \times 100 \times 100 \times 100)$  if the subcube had every possible intersection populated, it would be 100,000,000,000. It is easy to see how you could reach a memory limitation doing this.

You can measure the size of these subcubes. First, remember that each record is about 200 bytes for a 12-month application. Then, by using the rules in HFM, you can open a subcube and calculate the number of records.

HFM stores data for these dimensions in three sets of tables (these sets exist for each scenario-year combination per application):

- **DCE (Currency subcube)** Stores <Entity Currency>, <Entity Currency Adj>, <Parent Currency>, and <Parent Currency Adj>. It is also possible for a user to force a translation into another currency triplet, and this is stored here as well.
- **DCN (Parent subcube)** Stores other Value dimension members, the members in the Node. Any value in the Node relates to that specific parent-child relationship, and both the parent and child need to be



identified. This is called out by the additional fields in the tables that identify the parent. So, DCN tables are just like DCE tables but include an additional field for the parent.

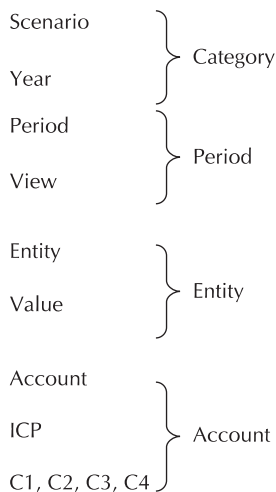
- **DCT (Journal transactions)** Stores all journal transactions, they transfer data values to DCE (for <Entity Currency Adjs> and <Parent Currency Adjs>) or DCN tables (for [Parent Adjs] and [Contribution Adjs]) when posted.

There are settings you could modify; however, for most applications you will never need to change them. The default settings work very well. HFM uses an algorithm to identify the least recently used (LRU) cubes and purges them from memory. If you do decide you need to modify these settings, you need to find a balance in the size of the LRU and the amount of memory assigned for data cache. For example, too large an LRU means you hold too many records in memory, putting the system under more memory pressure, which reduces system performance. It is best to consult an expert about these settings, or perform these modifications in a test environment or lab.

## Migrating Your Application from Hyperion Enterprise

If you are one of Oracle's long-time customers who have been looking at HFM, you should understand how the dimensions align, and what tools are available for migrating to HFM from Hyperion Enterprise (HE). One of the primary reasons for upgrading from HE to HFM is to gain an increase in dimensionality. There are many similarities from HE to HFM, but HFM offers you 12 dimensions, and HE only 4. In fact, any updates to HE dimensionality mean that you are adding new members to either accounts or entities. This is a technical limitation of a flat-file database, on which Enterprise is based. Figure 1-20 shows how these dimensions align from HE to HFM.

You can see that HE dimensions are aligned into a Hyperion Financial Management application. The Category dimension becomes Scenario and Year. Hyperion Enterprise accounts and subaccounts become the Account, ICP, and Custom dimensions 1–4. Lastly, HFM handles the Enterprise Period dimension using Period and Frequency (for example, MTD, YTD).



**FIGURE 1-20.** *Hyperion Enterprise dimensions aligning to Hyperion Financial Management dimensions*

Hyperion Enterprise has a utility to help you migrate your HE application to HFM. It is called Oracle Hyperion Enterprise Extraction Utility. The utility is designed to convert full applications, not just the metadata. First, you may need to upgrade to the most recent version of Enterprise (6.5.1). Then you extract the HFM files from that application. It does work and can save you some time. It produces the application files for you to either update or just load. It is useful in that it brings over the Accounts and Entities and descriptions. Although the utility attempts to place them into hierarchies, this is one place that will need some updating. How well it can do this has a lot to do with the Enterprise structure you start with.

This utility will not do all the work for you, but it is part of HE and will save you some time. The best approach is to redesign the application to take advantage of features offered in the consolidation products.

## Conclusion

At this point in a typical project, you should really see some great progress. Following this chapter, you should have a good design, skilled team, strong consulting partner, and plan—you have ensured that you will be ready for the next phase of the project. And most importantly, you have a strong foundation to ensure your success.