Taming the Big Data Tidal Wave: Finding Opportunities in Huge Data Streams with Advanced Analytics
By Bill Franks

CHAPTER 5

The Evolution of Analytic Processes

What does the increased scalability discussed in Chapter 4 buy an organization? Not much, if it isn't put to use. Upgrading technologies to today's scalable options won't provide a lot of value if the same old analytical processes remain in place. It will be a lot like buying a new 3-D TV with all the bells and whistles, and then simply connecting it to an antenna, grabbing local TV signals from the air. The picture might be improved over your old TV, but you certainly won't be changing your viewing experience very much compared to what is possible with the new TV.

Similarly, as the options practitioners of advanced analytics have in terms of scalability evolve, so, too, must the processes used to generate and deploy analytics. Legacy processes for deploying analytic routines simply aren't able to take advantage of the current state of the world. Without changing key aspects of existing analytical processes, organizations will not realize more than a fraction of the gains in power and productivity that are possible with the new levels of scalability available today. It isn't possible to tame big data using only traditional approaches to developing analytical processes.

One process that needs to be changed is the process of configuring and maintaining workspace for analytic professionals. Traditionally, this workspace was on a separate server dedicated to analytical

processing. As already discussed, in-database processing is becoming the new standard. However, in order to take advantage of the scalable in-database approach, it is necessary for analysts to have a workspace, or "sandbox," residing directly within the database system. In the big data world, a MapReduce environment will often be an addition to the traditional sandbox. The first part of this chapter will discuss what an analytical sandbox is, why it is important, and how to use it.

As analytic professionals begin consistently leveraging a database platform for their work through a sandbox, they will be doing some tasks repeatedly. For example, each individual analyst will need certain core metrics on customers regardless of the specific analysis he or she is building. Enterprise analytic data sets are key tools to help drive consistency and productivity, and lower risk into an organization's advanced analytics processes. The second part of the chapter will start with an overview of what a basic analytic data set is. Then, we'll cover what an enterprise analytic data set (EADS) is, what the benefits of an EADS are, and how an EADS can be utilized by other users or applications beyond the analytic professionals it is developed for.

Many analyses lead to "scoring routines" that need to be deployed and applied on a regular basis. For example, a customer propensity model scoring routine might provide the likelihood that any given customer will purchase a certain product in the next month. Historically, updating scores for each customer was a time-consuming and infrequent task. In today's world, it is often necessary to keep scores up to date on a daily, if not real-time, basis. The third part of the chapter will discuss how to embed scoring routines within the database environment, as well as how to effectively track and manage the models and processes that are developed through model management.

THE ANALYTIC SANDBOX

In Chapter 4, we discussed the power of massively parallel database systems. One of the uses of such a database system is to facilitate the building and deployment of advanced analytic processes. In order for analytic professionals to utilize an enterprise data warehouse or data mart more effectively, however, they need the correct permissions and

access to do so. An analytic sandbox is the mechanism for achieving this. If used appropriately, an analytic sandbox can be one of the primary drivers of value in the world of big data.

The term "sandbox" originates from the sandboxes that many children play in. Within a sandbox, children can create anything they like. They can reshape the sand at will, depending on their desires at the time. Similarly, a sandbox in the analytics context is a set of resources that enable analytic professionals to experiment and reshape data in whatever fashion they need to. Other terms used for the sandbox concept include an agile analytics cloud and a data lab, among others. Which term you apply to the concept isn't important. What is important is that you embrace the concept.

The Analytic Sandbox: Definition and Scope

An analytic sandbox provides a set of resources with which in-depth analysis can be done to answer critical business questions. An analytic sandbox is ideal for data exploration, development of analytical processes, proof of concepts, and prototyping. Once things progress into ongoing, user-managed processes or production processes, then the sandbox should not be involved.

A sandbox is going to be leveraged by a fairly small set of users. There will be data created within the sandbox that is segregated from the production database. Sandbox users will also be allowed to load data of their own for brief time periods as part of a project, even if that data is not part of the official enterprise data model.

Data in a sandbox will have a limited shelf life. The idea isn't to build up a bunch of permanent data. During a project, build the data needed for the project. When that project is done, delete the data. If used appropriately, a sandbox has the capability to be a major driver of analytic value for an organization.

Analytic Sandbox Benefits

What are the benefits of an analytic sandbox? Let's explore the topic from both an analytic professional's view and IT's view.

Benefits from the view of an analytic professional:

- **Independence.** Analytic professionals will be able to work independently on the database system without needing to continually go back and ask for permissions for specific projects.
- **Flexibility.** Analytic professionals will have the flexibility to use whatever business intelligence, statistical analysis, or visualization tools that they need to use.
- **Efficiency.** Analytic professionals will be able to leverage the existing enterprise data warehouse or data mart, without having to move or migrate data.
- **Freedom.** Analytic professionals can reduce focus on the administration of systems and babysitting of production processes by shifting those maintenance tasks to IT.
- **Speed.** Massive speed improvement will be realized with the move to parallel processing. This also enables rapid iteration and the ability to "fail fast" and take more risks to innovate.

A SANDBOX IS GOOD FOR EVERYONE!

There are distinct advantages tied to a sandbox environment for both analytic professionals and IT. It isn't something that hurts one group at the expense of the other. People on both sides are often afraid of the concept at first since they don't understand it. It may take some time to educate people and get past that initial reaction. The effort will be worth it.

Benefits from the view of IT:

- **Centralization.** IT will be able to centrally manage a sandbox environment just as every other database environment on the system is managed.
- **Streamlining.** A sandbox will greatly simplify the promotion of analytic processes into production since there will be a consistent platform for both development and deployment.
- **Simplicity.** There will be no more processes built during development that have to be totally rewritten to run in the production environment.

- **Control.** IT will be able to control the sandbox environment, balancing sandbox needs and the needs of other users. The production environment is safe from an experiment gone wrong in the sandbox.
- **Costs:** Big cost savings can be realized by consolidating many analytic data marts into one central system.

An Internal Sandbox

For an internal sandbox, a portion of an enterprise data warehouse or data mart is carved out to serve as the analytic sandbox. In this case, the sandbox is physically located on the production system. However, the sandbox database itself is not a part of the production database. The sandbox is a separate database container within the system. See Figure 5.1 for an illustration.

Note that with big data, it will be wise to also add a MapReduce environment into the mix. This would typically be installed alongside the database platform unless you're using a system that can combine the two environments together. The MapReduce environment will require access to the internal sandbox. Data can be shared between the two environments as required. We talked about MapReduce in Chapter 4.

One strength of an internal sandbox is that it will leverage existing hardware resources and infrastructure already in place. This makes it very easy to set up. From an administration perspective, there's no difference in setting up a sandbox than in setting up any other database container on the system. What's different about the sandbox are some of the permissions that will be granted to its users and how it is used.

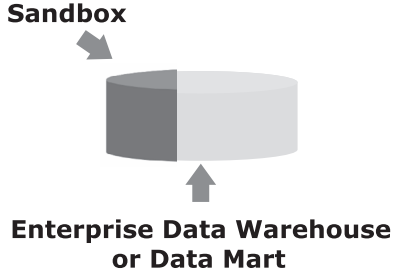

Figure 5.1 An Internal Sandbox

Perhaps the biggest strength of an internal sandbox is the ability to directly join production data with sandbox data. Since all of the production data and all of the sandbox data are within the production system, it's very easy to link those sources to one another and work with all the data together. See Figure 5.2 for an illustration of how this is possible.

An internal sandbox is very cost-effective since no new hardware is needed. The production system is already in place. It is just being used in a new way. The elimination of any and all cross-platform data movement also lowers costs. The one exception is any data movement required between the database and the MapReduce environment.

There are a few weaknesses of an internal sandbox. One such weakness is that there will be an additional load on the existing enterprise data warehouse or data mart. The sandbox will use both space and CPU resources (potentially a lot of resources). Another weakness is that an internal sandbox can be constrained by production policies and procedures. For example, if on Monday morning virtually all the system resources are needed for Monday morning reports, sandbox users may not have many resources available to them.

An External Sandbox

For an external sandbox, a physically separate analytic sandbox is created for testing and development of analytic processes. It's relatively rare to have an environment that's purely external. Internal or

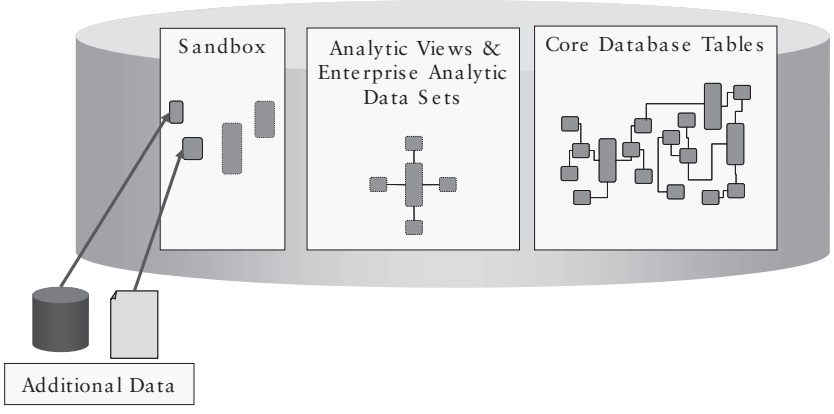

Figure 5.2 Detailed Internal Sandbox View

hybrid sandboxes, which we'll talk about next, are more common. It is important to understand what the external sandbox is, however, as it is a component of a hybrid sandbox environment. See Figure 5.3 for an illustration.

The biggest strength of an external sandbox is its simplicity. The sandbox is a stand-alone environment, dedicated to advanced analytics development. It will have no impact on other processes, which allows for flexibility in design and usage. For example, different database settings can be explored or an upgrade to a newer version of the database can be done to test new features. This is similar to what is often done on traditional test and development systems used for building applications.

One common question that often arises is "Isn't this external system completely violating this concept of keeping the data in-database when analyzing it?" The answer is no if you consider it an analytics development environment. Most organizations have a test and/or development environment, independent of their production system, for application and business intelligence work. It's a necessary component to help build, test, and debug new processes. An external sandbox is exactly the same concept for the exact same reasons, only it's dedicated to analytic initiatives.

Another strength of an external sandbox is reduced workload management. When only analytic professionals are using the system, it isn't necessary to worry much about tuning and balancing. There will be predictable, stable performance in both the sandbox and production environments. For example, sandbox users won't have a Monday morning downgrade to their resources due to reporting needs. They'll have a steady level of access to the sandbox.

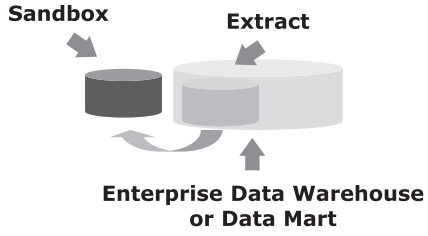

Figure 5.3 An External Sandbox

AN EXTERNAL SANDBOX DOESN'T VIOLATE THE RULES

An external sandbox does not violate the rules of in-database processing. View an external sandbox as a test and development environment for analytics. There are many valid and compelling justifications for such environments, and they are ubiquitous for application and report development.

An external sandbox is preferably a relational database of the exact same nature as the production system. This way, moving processes from the sandbox to the production environment is simply a matter of copying things over. If data extracts sent to the sandbox are kept in the same structure as on production, migrating will be easy to do.

When it comes to working with big data, a MapReduce environment should be included as part of an external sandbox environment. In this case, the external sandbox environment will have a relational database and a MapReduce component. In some cases, one system can handle both responsibilities, in other cases there will be two physical platforms required.

A major weakness of an external sandbox is the additional cost of the stand-alone system that serves as the sandbox platform. To mitigate these costs, many organizations will take older equipment and shift it to the sandbox environment when they upgrade their production systems. This makes use of equipment that would otherwise be discarded and saves any costs associated with hardware for the sandbox.

Another weakness is that there will be some data movement. It will be necessary to move data from the production system into the sandbox before developing a new analysis. The data feeds will also need to be maintained. The feeds don't have to be too complicated, but it is an extra set of tasks to maintain and execute. Any data feeds should be scoped very tightly and should focus only on what is absolutely needed.

A Hybrid Sandbox

A hybrid sandbox environment is the combination of an internal sandbox and an external sandbox. It allows analytic professionals the

flexibility to use the power of the production system when needed, but also the flexibility of the external system for deep exploration or tasks that aren't as friendly to the database. See Figure 5.4 for an illustration.

The strengths of a hybrid sandbox environment are similar to the strengths of the internal and external options, plus having ultimate flexibility in the approach taken for an analysis. It is easy to avoid production impacts during early testing if work is done on the external sandbox. When it comes time for final testing and pre-deployment work, the production sandbox can be used. A single MapReduce environment might augment the hybrid sandbox by supporting both the internal and external sandboxes.

Another advantage is if an analytic process has been built and it has to be run in a "pseudo-production" mode temporarily while the full production system process is being deployed. Such processes can be run out of the internal sandbox easily.

The weaknesses of a hybrid environment are similar to the weaknesses of the other two options, but with a few additions. One weakness is the need to maintain both an internal and external sandbox environment. Not only will it be necessary to keep the external sandbox consistent with the production environment in this case, but the external sandbox will also need to be kept consistent with the internal sandbox.

It will also be necessary to establish some guidelines on when each sandbox option is used. There ought to be certain types of activities that are earmarked for the external sandbox and certain activities earmarked for the internal sandbox. It can't be a matter of analytic

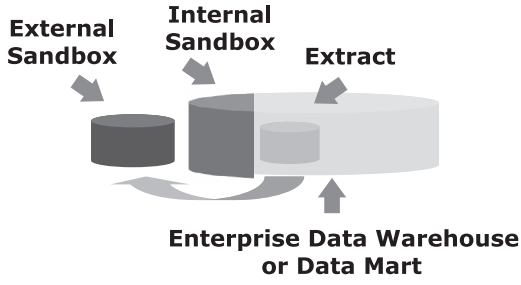

Figure 5.4 A Hybrid Sandbox

professionals arbitrarily using one or the other. The analytics team is going to have to develop guidelines and stick to them.

DON'T OVERFILL A SANDBOX

Only the minimum data needed for analysis efforts should be copied to an external sandbox environment. The sandbox should host only a small portion of the data stored in the production environment. The specific data will change over time depending on current analytical needs. Don't replicate more than absolutely necessary.

The last weakness is that some two-way data feeds may be required, which adds complexity. There must be some consistency in the data available to the internal sandbox and the external sandbox. As new data is developed in one of the sandbox environments, it may need to be replicated it in the other.

Don't Just Use Data. Evolve It!

One of the best uses for a sandbox environment is to identify new data sources that should be added into an organization's systems and processes on an ongoing basis. Perhaps you purchase a feed of social media data, or a household demographic file, or you get a feed of a new big data source. How will analytic professionals explore and experiment with this new data?

Imagine the folly of the typical approach of setting up a process to formally get the new data in production before it can be explored. What would that entail? You'll have to justify and scope the project to load the data. Then there will be a project commissioned to develop the extract, transform, and load (ETL) processes to load the data into the system. A data model to put the data into will need to be designed, approved, and implemented. Then all of the above must be tested. After three or six months, the process is turned on, and the data is in there and ready for use. At that point, perhaps analysis shows the data doesn't have much value and you don't really need it. What a waste of resources it was to get it formally added to the system!

TASTE A SAMPLE FIRST

When people aren't sure they'll like an ice cream flavor, they ask to try a taste. If they like it, they order a whole serving. If not, they move on to another flavor. Follow the same logic with new data sources, especially big data sources. Don't go for a whole serving until you know it is something you want. Experiment and try the data first in your sandbox.

The way a sandbox is used to avoid the preceding scenario is to get a onetime feed of the new data, load it up in the sandbox, and test it out. If it doesn't work out very well, move on. If it does, that is the time to kick off the lengthy and expensive process to get the data formally loaded on an ongoing basis. Using an analytical sandbox to explore and prove the value of new data sources is a vastly faster and cheaper way to go than traditional paths.

Workload Management and Capacity Planning

As analytic professionals start to use a sandbox, there are a lot of built-in components of database systems that will enable it to work smoothly. Sandbox users can be assigned to a group that has permissions that make sense for the purpose of developing new advanced analytics processes. For example, it is possible to limit how much of the CPU a given sandbox user can absorb at one time. Enterprise class systems are flexible enough to let users get only 10 percent of resources if the system is in demand, but if it is nighttime and no one else is active, a user can use the entire system.

It is possible to control the number of concurrent queries or even the types of queries that users can make. Perhaps individuals are only allowed five concurrent jobs at a time. There can also be processes to identify poorly formed queries, such as a query that includes a cross-join of two large tables, and terminate such queries.

One of the important things to do is to limit disk space usage through data retention policies. When a data set is in a sandbox and it hasn't been touched in a couple of months, the default should be

that it is deleted. A sandbox should not just continuously build up data sets, as often happens in traditional environments.

I've literally seen clients who have five terabytes of distinct corporate data, yet their analytic environment has 30 to 50 terabytes. The reason is that each analyst has made a copy of much of the five terabytes. Each analyst may even have multiple copies for different projects. This causes a huge proliferation of replicated, redundant data. That same approach should not be replicated within the sandbox environment. Within the sandbox, unless someone requests something be kept for a specific purpose, it needs to be deleted.

Especially with an internal sandbox, as more analytics are implemented, it will change the mix and level of resource usage in both the sandbox environment and the production environment. That's okay. Since the environments are on a single standardized platform, then the analytic processing can be accounted for in projections of usage just the same as everything else. Capacity plans should be discussed before getting started, but there is nothing special about sandbox processing that changes what the people who develop capacity plans need to do. Sandbox activity simply folds in. The system administrators know how to take care of this.

THE OPPOSITE OF WHAT MANY ASSUME IS TRUE!

A sandbox environment can drive additional value from current investments rather than adding costs. It doesn't inherently cause the need for new equipment. Nor does a sandbox inherently cause problems with other processes. It can drive more value from existing investments without any negative impacts. Once you understand a sandbox and how it works, you will find that the opposite of what many assume is true!

There's one really big, common misnomer. People often think that an analytic sandbox is going to "destroy" the system, use up all the resources, and cause a lot of mayhem. That's not true at all. In fact, very big analytic jobs usually need to be run once or twice at the beginning of a project. They are not being run again and again. Big jobs can be easily scheduled to run at night, for example, when the

system would otherwise be underutilized. As opposed to the use of an analytic sandbox somehow eating up all system resources, and pushing it over a cliff, it can actually lead to the opposite. The analytics being run in a sandbox can use resources that would otherwise be sitting idle. This helps drive even more value out of the investments made in the infrastructure without necessitating any increases to it. That's a great thing!

This leads to a final, related point. Adding analytics into an environment with a sandbox does not inherently, in and of itself, require any new capacity. If a system is 95 or 99 percent utilized today, then putting an internal sandbox on it is probably going to require upgrading that system. That's not an inherent impact of the analytic sandbox, however. That's a function of the fact that the system is so busy that any new application or process placed on it is going to require adding more capacity. Similarly, if older equipment is used for an external sandbox, there are no new costs. In fact, new value will be driven from equipment that would have been thrown away and not producing any value at all.

WHAT IS AN ANALYTIC DATA SET?

An analytic data set (ADS) is the data that is pulled together in order to create an analysis or model. It is data in the format required for the specific analysis at hand. An ADS is generated by transforming, aggregating, and combining data. It is going to mimic a denormalized, or flat file, structure. What this means is that there will be one record per customer, location, product, or whatever type of entity is being analyzed. The analytic data set helps to bridge the gap between efficient storage and ease of use.

Most data in relational databases is stored in what is known as third normal form. This is a method of storing data that eliminates data redundancy but makes queries more complex. Third normal form table structures are very efficient for storing and retrieving data, but they cannot be directly used for most advanced analytics efforts. Getting into the details of what third normal form is all about is out of the scope of this book. What matters is that analytic tools usually require data in a simple, denormalized, flat file format. The

sophistication in advanced analytics is in the algorithms and methods applied to the data, not in the structure of the data itself. Such data sets can take several forms, which we'll discuss next.

Development versus Production Analytic Data Sets

There are two primary kinds of analytic data sets, which Figure 5.5 illustrates. A development ADS is going to be the data set used to build an analytic process. It will have all the candidate variables that may be needed to solve a problem and will be very wide. A development ADS might have hundreds or even thousands of variables or metrics within it. However, it's also fairly shallow, meaning that many times development work can be done on just a sample of data. This makes a development ADS very wide but not very deep.

A production analytic data set, however, is what is needed for scoring and deployment. It's going to contain only the specific metrics that were actually in the final solution. Typically, most processes only need a small fraction of the metrics explored during development. A big difference here is that the scores need to be applied to every entity, not just a sample. Every customer, every location, every product will

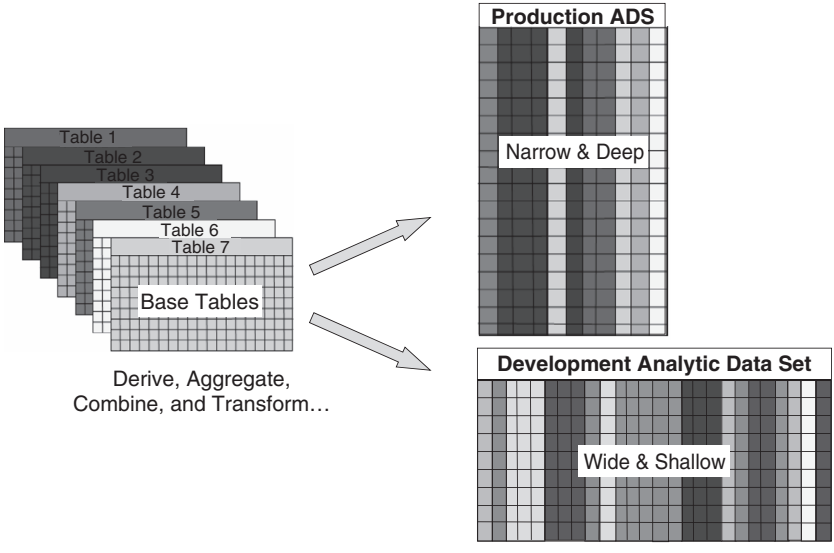

Figure 5.5 Development versus Production Analytic Data Sets

need to be scored. Therefore, a production ADS is not going to be very wide, but it will be very deep.

For example, when developing a customer model, an analytic professional might explore 500 candidate metrics for a sample of 100,000 customers. The development ADS is therefore wide but shallow. When it comes time to apply scores to customers in production, perhaps only 12 metrics are needed but they are needed for all 30,000,000 customers. The production ADS is therefore narrow but deep.

Traditional Analytic Data Sets

In a traditional environment, all analytic data sets are created outside of the database, as illustrated in Figure 5.6. Each analytic professional creates his or her own analytic data sets independently. This is done by every analytic professional, which means that there are possibly hundreds of people generating their own independent views of corporate data. It gets worse! An ADS is usually generated from scratch for each individual project. The problem is not just that each analytic professional has a single copy the production data. Each analytic professional often makes a new ADS, and therefore a new copy of the data, for every project.

As mentioned earlier, there are cases where companies with a given amount of data end up with 10 or 20 times that much data in their analytic environment. As an organization migrates to a modern, scalable process, it doesn't want to carry over the model of having all of these different copies of the data for each of the users. An alternative method is required, which we'll get to in a moment.

One of the big issues people don't think about with traditional ADS processes is the risk of inconsistencies. Perhaps an analytic professional has ended up over time defining sales as gross sales net of returns and discounts. At the same time, the person in the next cube has been defining sales as gross sales net of discounts, but not returns. It is not that either of them is wrong in any absolute sense, but each has a slightly different definition. If they are doing work for the same business partners, they will have inconsistencies in how they do their analysis and report their results back to their partners. This can cause problems.

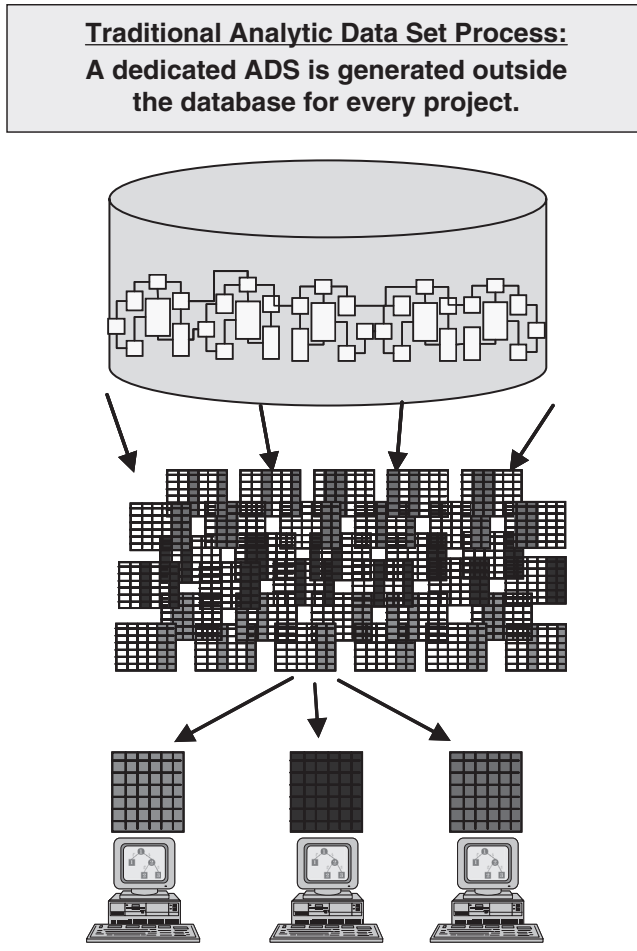

Figure 5.6 Traditional Analytic Data Set Process

Another huge issue with the traditional approach to analytic data set generation is the repetitious work. If analytic professionals are creating very similar data sets again and again, it's not just the space and system resources they are using, but it's their time. They have to set up the ADS processes, they have to run them, and they have to babysit them and make sure they are complete. This extends all of their time lines and adds costs to all of their projects.

INCONSISTENCY CAN HURT MORE THAN REDUNDANCY

While it is true that a traditional analytic data set generation process leads to a lot of redundant data, this isn't the biggest problem. Often overlooked is the fact that different analytic professionals may develop slightly different definitions of key metrics over time. Their results will therefore be inconsistent. These inconsistencies are often not acknowledged or even known.

There is one last area of wasted energy and resources. Once an ADS process is created for a project, the work is not done. When the process is ready for production, an analytic professional is going to have to reverse engineer it to push it back up into production. If, as is typical historically, the production environment is not the same platform as the development environment, that will involve rewriting the entire process to work in the production environment. For example, it may be necessary to translate code from an analytic tool into SQL or a UDF. This is a very costly and error-prone step. Some companies spend more time and money deploying a ready-to-go analytic process than they did creating the process to begin with!

ENTERPRISE ANALYTIC DATA SETS

Let's talk about a way to streamline the analytic data set creation process through an enterprise analytic data set, or EADS. An EADS is a shared and reusable set of centralized, standardized analytic data sets for use in analytics.

What an EADS does is to condense hundreds or thousands of variables into a handful of tables and views. These tables and views will be available to all analytic professionals, applications, and users. The structure of an EADS can be literally one wide table, or it may be a number of tables that can be joined together.

An EADS is collaborative in that all of the various analytic processes can share the same, consistent set of metrics. An EADS is going to greatly simplify access to data by making many metrics available

directly to analytic professionals without further effort. They no longer have to go and navigate the raw third normal form tables and derive all the metrics themselves. An EADS is going to greatly reduce time to results and it is a "build once, use many" endeavor. See Figure 5.7 for an illustration.

One of the most important benefits of an EADS, which isn't often the first that people think about, is the consistency across analytic efforts. With greater consistency in the metrics feeding an organization's analytics, people can be comfortable that metrics feeding different processes were computed identically. Enterprise analytic data sets, if used appropriately, can help reduce data preparation time from 60 to 80 percent of total project effort to a much lower percentage. A

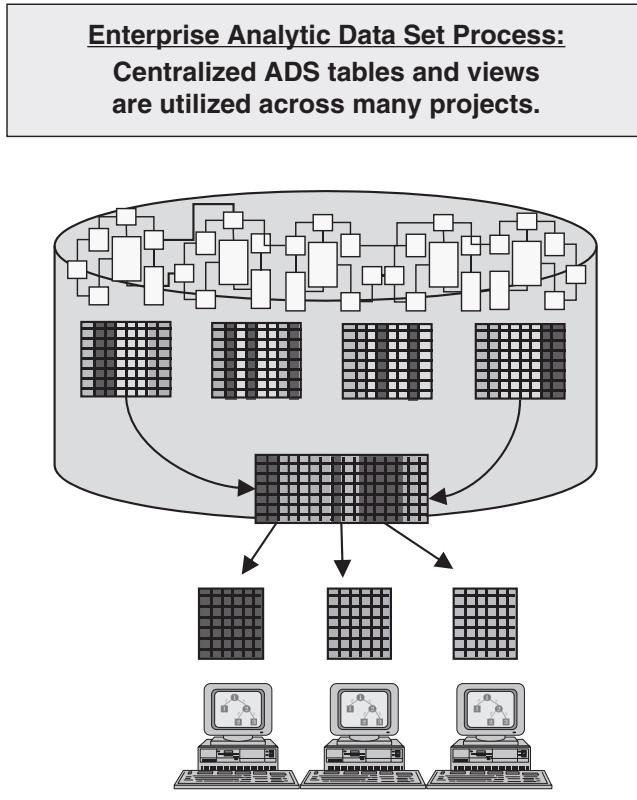

Figure 5.7 Enterprise Analytic Data Set Process

goal might be to get as low as 20 to 30 percent. Key features of an enterprise analytic data set include:

- A standardized view of data to support multiple analysis efforts.
- A method to greatly streamline the data preparation process.
- A way to provide greater consistency, accuracy, and visibility to analytics processes.
- A way to open new views of data to applications and users outside of the advanced analytics space.
- Something that will allow analytic professionals to spend much more time on analysis!

When to Create an Enterprise Analytic Data Set

When should you create an enterprise analytic data set? You should create an EADS when you're doing a lot of analysis in a given area and you expect to do a lot more of it. You can create an EADS for any entity on which you focus your analysis. Such entities include customers, products, locations, employees, and suppliers. Anything analyzed on a regular basis is a candidate for an EADS. An EADS will evolve over time. If a new source of big data is identified, additional metrics can be added to an EADS to account for the new information.

There will be time and cost associated with creating enterprise analytic data sets. Don't let that scare you. The costs will be more than made up over time through the cost savings of both man-hours and equipment. The commodity modeling concept discussed in Chapter 6 is a perfect example of a type of analysis that would not be feasible without an enterprise analytic data set to support it.

It requires a cross-functional team to build an effective enterprise analytic data set. Business team members will have to help define the metrics that they want to study about the business. Analytics team members are going to build out the logic to generate those metrics. IT is going to have to maintain the enterprise analytic data set structures and processes within the environment where it will be deployed. Only by having all three teams contributing will the maximum benefit from the effort be realized. The next few sections will dig deeper into how the process works.

What Goes in an Enterprise Analytic Data Set?

Designing an enterprise analytic data set is actually a fairly simple process. The process starts by taking an inventory of all the metrics that are commonly used by all of the analytic professionals. If there are multiple versions of the same metric in use, incorporate them all. For example, perhaps some analytic professionals use total sales, net of discounts and returns, and some use total sales, net only of discounts. Include both versions in the EADS so that total gross sales net of discounts *and* total gross sales net of both returns and discounts are present. There is no need to choose just one! Over time, if more metrics are developed that are of importance, add them. An enterprise analytic data set should always be evolving. An EADS might start with a specific set of metrics today, but more metrics will be added over time.

CHOOSE THEM ALL!

There aren't many situations in life where you are presented with choices and can answer with "I'll take them all!" An enterprise analytic data set is one of those situations. Include every metric definition in use so that all needs are covered. The extra effort to include variations of a metric is trivial. Save the argument about which metrics are better for which uses for another day. You'll be covered no matter who wins that argument!

It is important to understand that the goal of an enterprise analytic data set is not to give analytic professionals 100 percent of the data required for every project. While it will hopefully approach 90 percent, any given project may need some custom metrics that aren't typically required and therefore aren't covered by the EADS. That's okay.

For example, perhaps deep analysis is desired on the top-selling products of the holiday season. An EADS likely contains data only at a product group level. In this case, a handful of specific products are the target of the analysis. Metrics for those products will have to be computed to augment the group-level data already present in the EADS. However, analyzing these individual products isn't a

common need, so it doesn't make sense to add the product metrics to the EADS.

If the enterprise analytic data set has 80 to 90 percent of needed metrics sitting there waiting, analytic professionals can focus on the extra 10 to 20 percent that need to be computed and then move on to their analysis. They can also leverage the logic utilized to create the EADS metrics. Over time, however, analytic professionals may see consistent, repetitious patterns in some of the extra metrics they are adding in. If so, then add those metrics to the EADS. There will need to be a governance process to handle adding the new metrics.

Logical versus Physical Structures

As we've discussed, an enterprise analytic data set is logically one row per entity, with dozens, hundreds, or thousands of metrics. If you are familiar with "old school" flat files, that's basically what we are talking about. Physically, as you can see in Figure 5.8, an EADS may not be stored in a way that matches that logical view.

While it may be logical to view a customer EADS as a customer table with sales, demographics, and direct marketing response information, it could very well be stored differently. The EADS might be

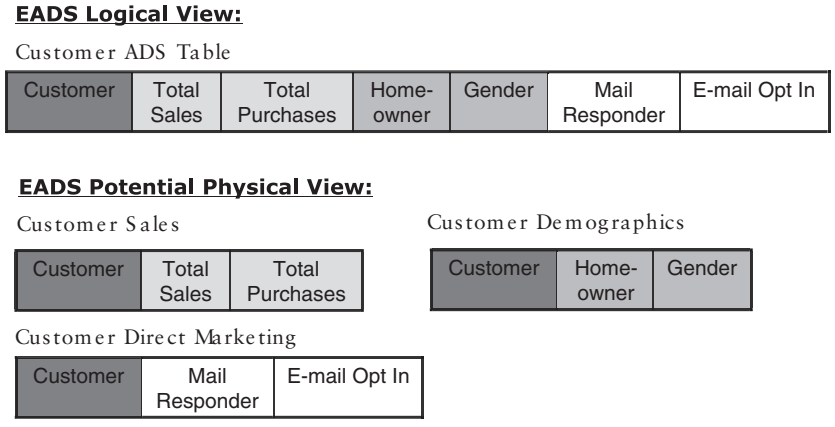

Figure 5.8 Logical versus Physical View of an EADS

physically stored so there is one table or view that has the sales metrics, one that has the demographic metrics, and one that has the direct marketing metrics.

Users don't have to worry about this. Once the right metrics are defined, the people who manage the database can figure out the ideal way to actually store it. Then, views can be added to provide users with the view they desire based on the physical tables.

Updating an Enterprise Analytic Data Set

Updating an enterprise analytic data set is one big driver of why there might be physically separate tables. Different types of data such as survey data, sales data, and demographics data can require updating with different frequencies. Maybe sales metrics need to be updated every day. Demographics may be updated quarterly. Survey information may never get updated once it is loaded. If a new survey is done, its data is loaded when the survey happens, but then it isn't touched again.

Based on that, it may be easier to have different types of data in different physical tables so that they can be updated independently. It will save system resources by not having the overhead of a lot of extra metrics in a table when updating only a few. In addition, by having those separate tables or views, it makes it easy for analytic professionals to go and pull the specific types of data that they want. Last, many databases have a limit to how many columns can be in one table. For a large EADS, multiple tables may be required to accommodate column limits if for no other reason.

Note that no matter how an EADS is physically stored, views can be placed on top to pull together different pieces as needed. One view might have just the sales and survey metrics, while another view has just the survey and demographic metrics, and another view has all three. Over time, if a new data source is added, such as social media data or web data, metrics based on it can be added into the enterprise analytic data set as well. An appropriate way to store the new data as well as an updated set of views to leverage the new data can be identified.

Summary Tables or Views?

One option for an enterprise data set is to have it be a set of summary tables that are updated via a scheduled process. There are benefits to a table-based enterprise analytic data set.

First, you have a true "compute once, use many" scenario. The total system load from analytic professionals is going to be vastly reduced. This is because instead of having each person repetitiously running the same type of process to do big joins and aggregations, it's going to be run one time in batch and then shared.

Another advantage is that most advanced analytics efforts involve a heavy use of historical data. Having slightly out-of-date data isn't going to make a big difference. Perhaps an organization is updating its EADS sales information nightly, or even just weekly. For most advanced analytics projects, that's fine. When many metrics are cumulative, they won't be impacted much either. For example, average market basket size by customer won't change much if a sale from today is missed given that there is a year of other history entering the computation.

A final benefit is that the analytic professionals are going to have very low latency in getting their data. Since the EADS tables are sitting there waiting, the analytic professionals can get right to it. No more waiting for big queries to run. They are going to be able to get straight to their analysis.

There are a few downsides to a table-based EADS. First is that the enterprise analytic data set tables will not be fully up-to-date with the latest data. Second is that they will use disk space on the system, potentially a whole lot of it. Last, an appropriate update schedule for the various components will have to be determined and appropriate processes put in place.

A second option is to make the enterprise analytic data set a series of views that are run on demand. There are several benefits to this approach.

First, the enterprise analytic data sets are always going to be completely fresh and updated. Next, if real-time or near-real-time analysis is important, analytic professionals will be fine since they're always getting the latest data. Last, if any changes are made to the enterprise data set, they will be immediately available. The second the views are

updated, anyone who next queries the view will get the updates included.

DO WHAT IS NEEDED

You'll need to decide how frequently to update your enterprise analytic data sets. You'll also need to decide whether to store your EADS as physical tables, logical views, or both. Let the facts guide you to the answer. The requirements gathered should make it clear what direction to go. In most cases, there will be a combination of tables and views.

Downsides to view-based enterprise analytic data sets exist as well. First, the system load won't necessarily be reduced that much. This is because of the fact that even though the analytic professionals are leveraging the same view, it's still going to be running the process each time that someone queries it. However, while the system load won't be reduced a lot, there is still the huge benefit of the consistency and transparency of the computations. Last, analytic professionals will have to wait longer to get their data back since the data will be generated from the detailed data on demand instead of being precomputed.

In many cases, it will make sense to have a combination of both tables and views included in EADS structures. Some of the data may need to be fully up-to-date, while other data can be a bit stale. Do what's appropriate for each specific data source. Deciding whether to use a table or a view should be based on the requirements, performance needs, and space constraints.

Also be sure to look for ways to limit storage when tables are used. Don't store ratios or other similarly derived metrics. Use a view on top of a base table to compute such things. For example, if an EADS has total sales and total transactions, there's no reason to store sales per transaction. Put a view on top that divides sales by number of transactions. It uses virtually no extra system resources to compute that on demand, but will save a lot of space.

Spread the Wealth!

Once an organization has deployed an enterprise analytic data set, it needs to make maximum use of it. The EADS should not be something

utilized only by analytic professionals. There is no reason why business intelligence and reporting environments, as well as their users, can't leverage the EADS structures as well. Why build a lot of logic to compute metrics within a reporting environment if the metrics are already available and waiting in an EADS?

Similarly, any application that would benefit from the contents of an EADS should leverage it. One common example is a CRM tool leveraging a customer EADS to facilitate segmentation efforts. In this case, customer metrics from an EADS are made available to the CRM application. Then, users are able to select customers based on the metrics without computing them within the CRM tool. Another example would be a call center application utilizing a customer EADS to provide customer metrics to call center reps. In this case, when a customer calls, the rep may see a variety of metrics about the customer on the screen. These metrics, such as recent purchases, can help the rep determine how to best handle the call.

The point is that there is a wealth of information compiled within an EADS. It will help eliminate redundant efforts, greatly increase transparency, and improve consistency. It will provide more speed and scalability. Equally important is the fact that an EADS opens up a wide range of information directly to other users and applications that would otherwise not have direct access to it.

EMBEDDED SCORING

Once an organization has an analytic sandbox set up and has implemented enterprise analytic data sets, it will be able to develop analytic processes and models more quickly and consistently. Analytic processes will also be much more scalable. What's next? How is value driven from the new processes to take the organization to the next level? One way is through embedded scoring processes that enable analytic results to be used.

Embedded scoring involves enabling scoring routines to run in the database so that users can leverage the models built in an effective, scalable fashion. Successfully implementing embedded scoring will include not just deploying each individual scoring routine, but also a process to manage and track the various scoring routines that are deployed. Note that a "score" can be something generated from a

predictive model, or it can be any other type of output from an analytic process.

Just to review, analytic processes often result in the outputting of a new piece of information. Examples include a customer's likelihood to purchase a product, the optimal price point for a product, or the expected lift in sales that a specific location will see during a promotion. When the analytics developed are applied with current data, this is called scoring. For example, before determining who to send an e-mail to, the likelihood of each customer to respond is needed based on the most current data available. The process of updating those likelihoods is the scoring process, and it should be as automated and streamlined as possible. Getting scoring processes embedded within a database environment leads to a number of benefits. Let's explore them.

First, scores run in batches will be available on demand. If regularly scheduled batch updates to a set of scores are done, then when a user needs to access a score it will be there waiting. It's also possible to do a batch update only when needed. For example, an organization may update the scores for the customers being added to a mail list only at the time the mail list is created.

Next, embedded scoring enables real-time scoring. This is especially important for situations such as web offers. If someone's on a site now, he must be scored based on what is known about him right now, including what he just did on the site, to get the right offer to him when he browses the next page. Similarly, perhaps someone is on the phone with a call center. As the customer has a conversation with a call center rep, the rep inputs any new information he or she has learned. The inputting of that information might warrant an update to the customer's score so the rep knows the right path to go down next.

Next, embedded scoring will abstract complexity from users. It's very easy for both individual users and applications to ask for a score. The system handles the heavy lifting. As a result, embedded scoring will make scores accessible to less technical people.

A final benefit is having all the models contained in a centralized repository so they are all in one place. If an inventory of models and scores created is kept through a model management process, it is pos-

sible to keep track of what has been created more easily. No longer will analytic professionals across an organization keep the scoring processes they create within their specific control. Rather, they will be managed centrally and deployed for wider use.

Embedded Scoring Integration

Once scoring routines are deployed, the scores generated can be utilized very easily by users and applications. CRM applications, for example, can point to segmentation and propensity scores. All that a user of the CRM application has to do is click in the CRM tool to access the scores. Operational applications can also leverage scores. For example, perhaps a model is predicting probable out of stocks based on sales rates. When a high-risk situation is found, it is necessary to send an alert to a local manager. Or perhaps an airline has models looking at the probability of delays based on the weather. Those scores are updated by flight and are passed to the application used to track and manage those delays. Any user can also directly access the scores via an ad hoc query.

ANALYTIC RESULTS MUST BE USED TO DRIVE VALUE

In order to benefit from the analytic processes it builds, an organization must use the results. Without making the use of the analytics easy, the organization won't leverage them to the full extent possible. Embedded scoring processes are critical to enabling ease of use, which leads to scores being utilized by a wide range of users and applications.

In Chapter 4, we talked about options for leveraging parallel database systems. Those same methods apply for developing embedded scoring processes.

- SQL, the native language of the database, is one option. This is especially true for models like decision trees, linear regression, or logistic regression. Even to hand code a scoring routine in SQL is fairly simple for such models.

- User-defined functions make things a little fancier. They truly embed a scoring routine in the database as a native function.
- Predictive modeling markup language (PMML), as discussed, is a way to build a model in one system and pass information about the model to another system. The information passed enables the receiving system to generate scoring code automatically.
- Finally, embedded processes allow analytic tools to run in the database directly so that no translation from the analytic tool's language into other languages is necessary at all.

Review Chapter 4 for more information on each of these options. The important thing for our purpose here is that all of the options apply when implementing embedded scoring processes.

Model and Score Management

There are four primary components required to effectively manage all of the analytic processes an enterprise develops. They can be seen visually in Figure 5.9. The components include analytic data set

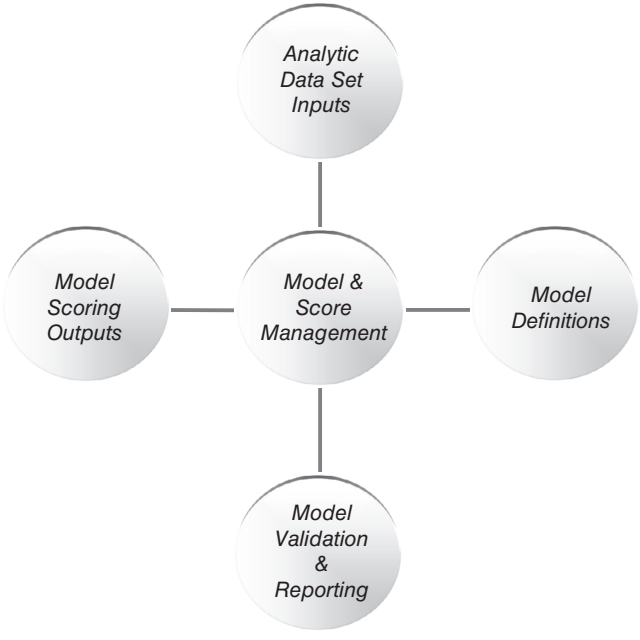

Figure 5.9 Model and Score Management Components

inputs, model definitions, model validation and reporting, and model scoring output. There are commercially available tools to help with model and score management, or a custom solution can be built to address an organization's specific needs. Let's review what each component is.

Analytic Data Set Inputs

It is necessary to track the details for each analytic data set or enterprise analytic data set that feeds into an analytics process. What needs to be tracked is a variety of information about the data sets, in addition to the technical facts needed to create and store them. This component of a model and scoring management system will manage information on the analytic data sets themselves. Note that these data sets may be enterprise analytic data sets, custom data sets for the process, or a combination of both. Information tracked includes:

- The name of the SQL script, stored procedure, user-defined function, embedded process, table, or view that will provide the data set to the user.
- The parameters that need to be entered to run the analytic data set process. Users might have to specify a date range or a product filter, for example.
- The output table(s) and/or view(s) that the process will create, along with the metrics they contain.
- The relationship between each analytic data set and the analytic processes that have been created. Any given analytic data set can feed into one or more models or processes. A given model or process might also require more than one analytic data set input.

Model Definitions

It is necessary to track a variety of information about each model or process. Note that a model in this case can be a true predictive model, or it can be some other analytic process, such as a ranking of customers by sales, that needs to be utilized on a regular basis. A model or process is registered with the model management system at the time it's created. Information tracked includes:

- The intended usage for the model. What business issue does it address? What are the appropriate business scenarios where it should be used?
- The history of the model. When was it created? Who created it? What revisions has it gone through?
- The status of the model. Is it still in development? Is it active and in production? Is it retired?
- The type of model. What algorithm was utilized? What methods were applied?
- The scoring function for the model. What is the name of the SQL script, stored procedure, embedded process, or user-defined function that will provide scores back to the user? Note that scoring functions assume that the required analytic data set tables are available.
- Information on the model input variables. What are the specific variables from the input analytic data set(s) that are used in the model or process? A given model or process might require metrics from just one ADS or it might require metrics from several.

Model Validation and Reporting

It is typically necessary to have a series of reports that help manage the models and processes over time. These reports can cover a range of topics and purposes. Information tracked includes:

- Reports that show how a specific run of scores compares to the development baselines.
- Specific summary statistics or validations, such as a lift or gains chart, that need to be reviewed after every scoring run.
- Model comparisons or variable distribution summaries.

Report output can be generated automatically when scores are updated or only upon request. Such reports are often used for the critical step of monitoring the performance of a model over time. All models will degrade as time passes and business situations evolve. Reports will help identify when it's time to revisit a model.

DON'T LOSE CONTROL

Without a concerted effort to track models and analytic processes, an organization runs the risk of models being used incorrectly due to confusion over what they are, or even models being totally forgotten about. A model and score management system will help ensure this does not happen. It also ensures that if one process is updated, it is easy to identify what other processes will be impacted.

Model Scoring Output

The final items it is necessary to track are the model scores that are output from the scoring process. These are the actual scores generated for every entity such as customer, location, or product. Information tracked includes:

- What is the score value? Where it is stored? What is the identifier of the customer, product, etc. that the score is for?
- The timestamp marking when a score was created.
- If desired, historical scores, as well as current scores. Some organizations keep historical scores for an extended period; others don't. In your organization, you can decide what makes sense.

WRAP-UP

The most important lessons to take away from this chapter are:

- Legacy processes for deploying analytical processes and models aren't designed to take advantage of the current state of the world. To tame big data, it is crucial that processes are updated to take full advantage of the scalability available.
- Analytic professionals need permissions beyond those that are typical. An analytic sandbox is a mechanism to give them the freedom they need while allowing IT to keep resources in balance.
- A sandbox is ideal for data exploration, analytic development, and prototyping. It should not be used for ongoing or production processes.

- There are several types of sandbox environments, including internal, external, and hybrid sandboxes. Each can be augmented with a MapReduce environment to help handle big data sources.
- An analytic data set (ADS) is a set of data at the level of analysis. Examples include customer, location, product, and supplier.
- Don't simply migrate traditional, project based ADS approaches to an in-database architecture. Instead, upgrade to a formal enterprise analytic data set (EADS) structure.
- An EADS is a set of predefined tables and views that provides easy access to hundreds or thousands of common metrics needed for analysis.
- An EADS improves performance, removes redundancy, increases transparency, and drives consistency across analytic initiatives.
- Open an EADS to applications and users in general, not just analytic professionals and analytic applications. An EADS has important information and should be shared widely.
- Embedded scoring builds on both the sandbox and EADS structures to provide scoring routines that are easily accessible by users and applications.
- Scoring can be embedded via SQL, user-defined function, embedded process, or PMML.
- Model and score management procedures will need to be in place to truly scale the use of models by an organization.
- The four primary components of a model and score management system are analytic data set inputs, model definitions, model validation and reporting, and model scoring output.