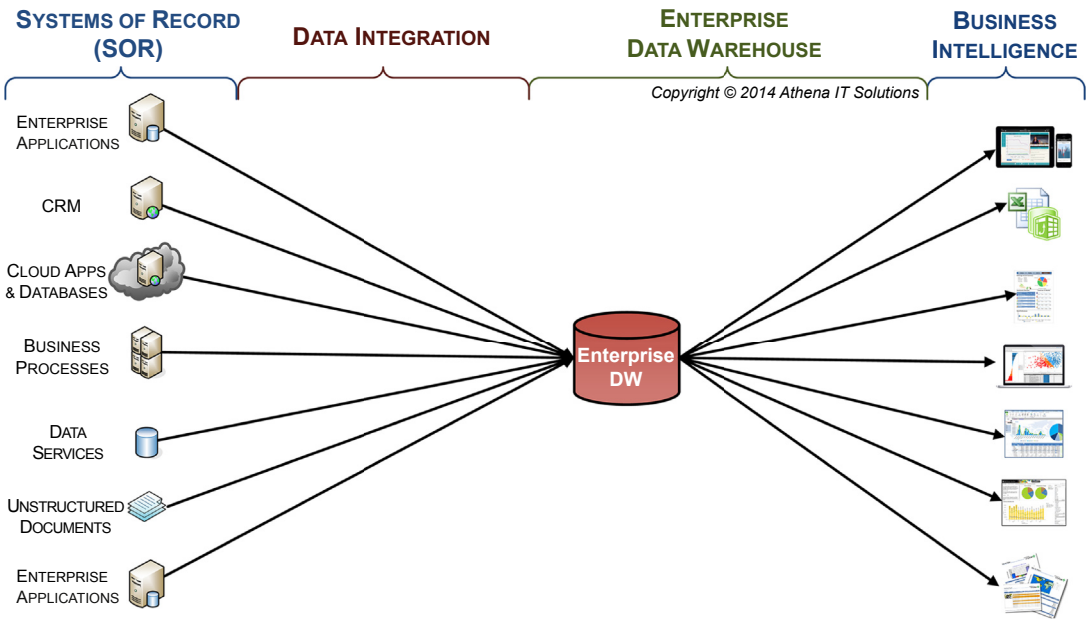

FIGURE 6.1

Enterprise data warehouse (EDW).

Designing, building, and implementing CDWs took a long time—months and maybe years. Some of the reasons for the lengthy time to deliver a CDW were:

- IT's use of the waterfall project methodology
- Everything had to be manually coded
- Much of the database, access, and networking standards that we take for granted today had not yet evolved, requiring the use of specific propriety APIs

The biggest reason why CDW projects took so long was IT's misguided attempt to gather all the requirements and implement an entire CDW in one "big bang" project. With an enterprise-wide scope, the project was destined to consume many resources, be late, and exceed the budget. An even worse outcome was that while the lengthy project was under way and after business requirements were documented, the business changed—invalidating many of the requirements.

Being an early adopter always carries some risk, and this was no exception. In addition to missing deadlines and running over budgets, CDW projects tended to under-deliver on expectations. And that was for the projects that actually were completed; many simply fizzled out. It is worth noting that many other big corporate projects in the 1990s encountered similar fates. There were scattered success stories, but, in general, the underwhelming results gave CDWs a bad reputation. It was inevitable, therefore, that there would be a backlash.

THE DATA MART

The next era was the rise of data marts. Data marts promised to be quicker and cheaper to build, and provided many more benefits—including the benefit of actually being able to finish building them! The data mart was

primarily a backlash to the big, cumbersome CDW projects, with the key difference being that its scope was limited to a single business group rather than the entire enterprise. Of course, that shortcut did speed things up, but at the expense of obtaining agreement on consistent data definitions, thereby guaranteeing data silos.

Although the early-adopter technology companies and Bill Inmon both advocated that data marts should be fed from the EDW, the rest of the industry was pushing data marts based on a different architecture where they were fed directly from source systems. Instead, data mart vendors and pundits stated there was no real architectural difference between a DW and data mart. As Figure 6.2 shows, the data mart, just like the DW, sourced data from SORs, used a waterfall project methodology and had to use quite a bit of manually-created custom code. The pundits' sales pitches said that the data mart was just like a DW – using the same tools, data models and design principles – but simpler and faster to build.

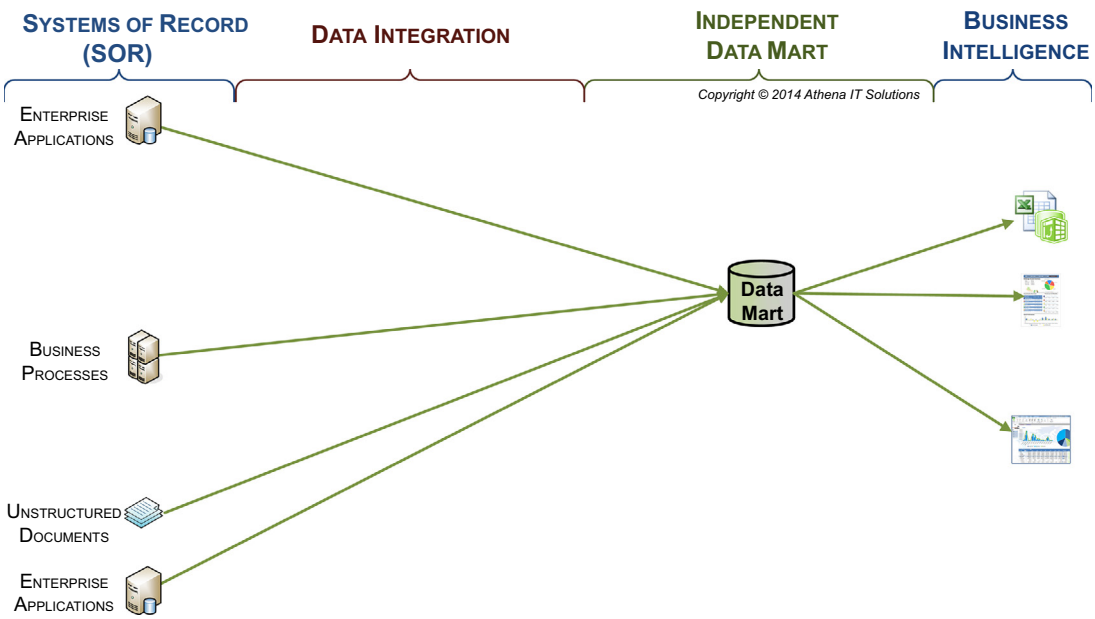

FIGURE 6.2

Data mart.

By the time data mart projects were starting to be widely built, two trends were also occurring: the emergence of BI tools and the use of relational databases to build new source systems. These trends resulted in a much shorter time to build a data mart than a CDW, aided, of course, by the fact that data marts have a more limited scope than CDWs.

Rather than being part of an overall enterprise data architecture, data marts were the architecture. You could build them faster and cheaper because you did not take the whole rest of the enterprise into account. This meant you could take shortcuts and avoid the most difficult part of DW: DI and reaching agreement on data definitions and metrics across business groups. As a result, data marts were built for the parochial interests of their sponsors rather than for the enterprise.

Companies sprouted multiple data marts, each built to accommodate only their sponsor with their own data definitions, data transformation, reporting, and technical platform. We moved from trying to

achieve one version of the truth to building multiple single versions of the truth—an oxymoron. To top it off, many of these projects overpromised and under-delivered, just like the CDW projects.

Data mart projects exacted another toll. The tradeoff for being able to deliver cheaper and faster was to severely limit scope of the breadth and depth of data, their integration, their quality, and the analytical capability offered to business users. Initially, data marts were perceived as a great success—until business groups realized they were debating in meetings which one of their reports (and associated data marts) had the "right" numbers. Each group had its own data silo.

In the end, data marts left their business users wanting more, which often meant building another round of data silos with more disjointed data marts or data shadow systems and more versions of the truth. At this point there started to be a debate within the industry on whether enterprises should build a DW versus data marts. Many vendors jumped on the data mart bandwagon because those projects were initially successful and they could therefore sell more of their products.

Later, this type of data mart would be become known as an independent data mart versus a dependent data mart that is part of a hub-and-spoke architecture. This meant it was independent from the DW and extracted data directly from the source systems rather than pulling the data from (and being dependent on) the DW.

MULTIPLE DATA MARTS

After the initial success of creating a data mart for a specific business group within an enterprise, more business groups wanted them and IT obliged. Each subsequent data mart was a success in its own right, at least initially, with each being built independently. As depicted in Figure 6.3 each data mart pulled data directly from the source systems and each report used one of the independent data marts as its data source.

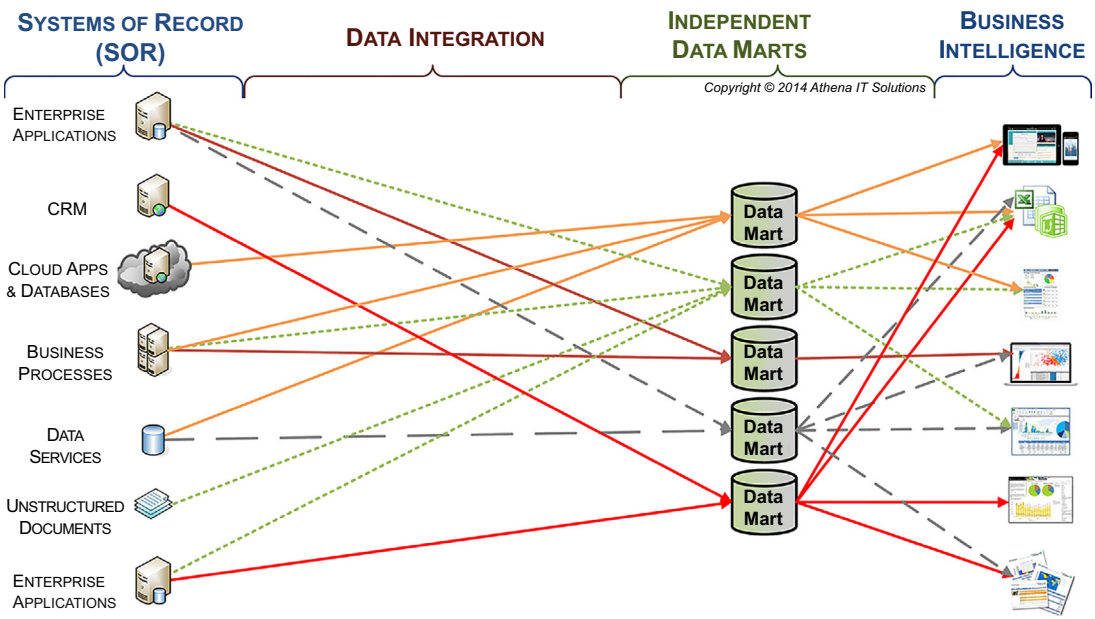

FIGURE 6.3

Multiple independent data marts.

After the independent data marts started proliferating in an enterprise, business people quickly started seeing differences in the reported business metrics they obtained from different data marts. The independent data marts became the new data silos for an enterprise. Each business group defined different business rules to filter, select, and transform the data in their data mart, causing greater disparity. Business people started debating the numbers in meetings and spent increasing amounts of time reconciling the data between reports that were generated from different independent data marts.

Although it was good that independent data marts could be built rather quickly and create business-group-specific reporting, it was at the expense of data consistency. It created new data silos with inconsistent reporting metrics across the enterprise. There is a phrase now about BYOD (bring your own device), but at that time the independent data marts (data silos) resulted in business groups debating BYOR (bring your own results).

In addition to the data silos created, there was a tendency for each data mart project to select its own tools (database, ETL, and BI) based on the myopic scope of a single business group. The result was an "accidental data architecture" for the enterprise. (The accidental architecture is discussed further in this chapter and in detail in Chapter 4.)

OPERATIONAL DATA STORE (ODS)

A new data architecture component called an ODS (see Figure 6.4) emerged in the early days of DW. The primary ODS objective was to bring data together from multiple source systems on as close to a

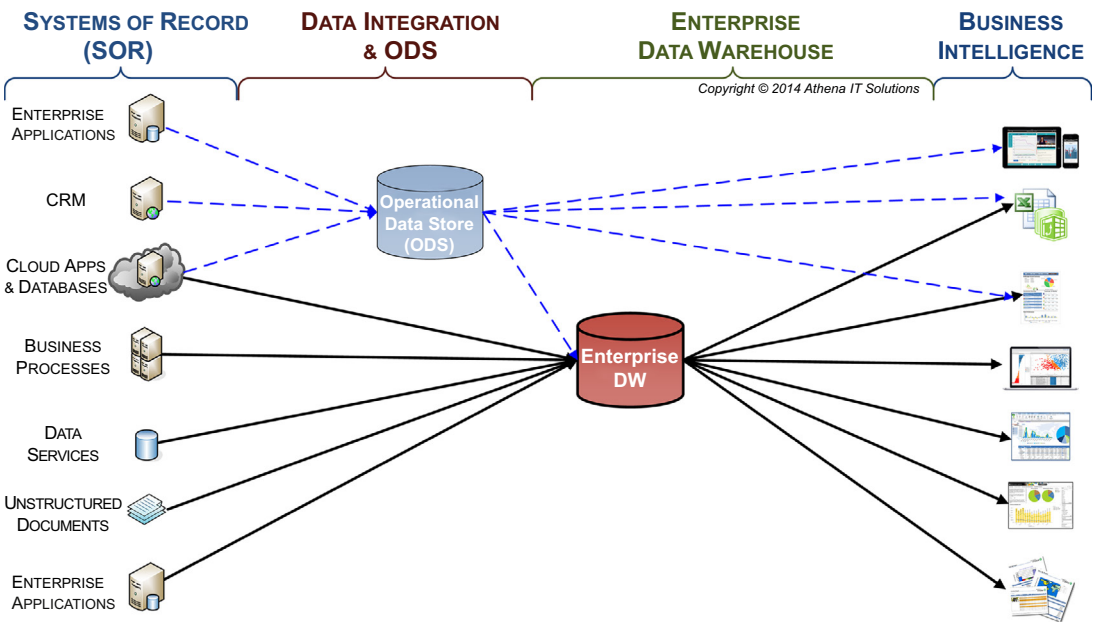

FIGURE 6.4

Operational data store.

real-time basis as possible to enable specific business processing or operational reporting. The typical ODS business requirements differed from a DW in these ways:

- Loading the data into one location, the ODS, for ease of access without any extensive DI or transformation. It was generally desirable for the data to remain as is rather than the usual cleansing, conforming, and transforming needed for a DW.
- Loading current data but generally not maintaining historical data.
- Loading application-specific data that supported business processes rather than a DW's business subject orientation.

The industry best practice was to use the ODS as the SOR to the EDW for the data it extracted from the source systems. The EDW would extract data from the source systems that the ODS did not use and would also extract data from the same source systems as the ODS if there was data needed by EDW that was not loaded into the ODS.

Enterprises with both an EDW and ODS could get their reports from:

- ODS only
- EDW only
- ODS and EDW

Although it was possible to manage reporting applications to avoid overlap and maintain integrity, just like ETL processing, it was all too common for the overlapping data sourcing options to create inconsistency.

An ODS may be a viable data architectural component; however, the architecture using an ODS and EDW without data marts suffered the same BI limitations as an EDW-only architecture such as not supporting robust BI; being inflexible with the inevitable changes in data and analytical requirements; and not adversely impacting IT and business productivity.

FEDERATED DWS

Federated DW is another architecture option that emerged. (See Figure 6.5) Federated DWs split the EDW into multiple physical DWs, using some of the following options to make the divisions:

1. Geographical regions or countries
2. Business functions such as finance, sales, marketing, or HR
3. Business entities such as divisions or subsidiaries

Although this data architecture was considered an alternative to the CDW by most people in the industry, the DW early adopters in the 1980s did not make a distinction between an EDW and a federated DW. The early adopters considered the EDW to be a logical entity that could be physically federated if it better supported that enterprise's BI requirements, and indeed the firm I worked in implemented that data architecture during that timeframe.

The best practice is to design EDW logically as a single data store, but physically implement it as either one data store or federated based on performance and business needs. Further, as I discuss in Chapter 7, when an enterprise is sourcing both structured and unstructured data into its BI architecture, it should use federated DWs based on the structures.

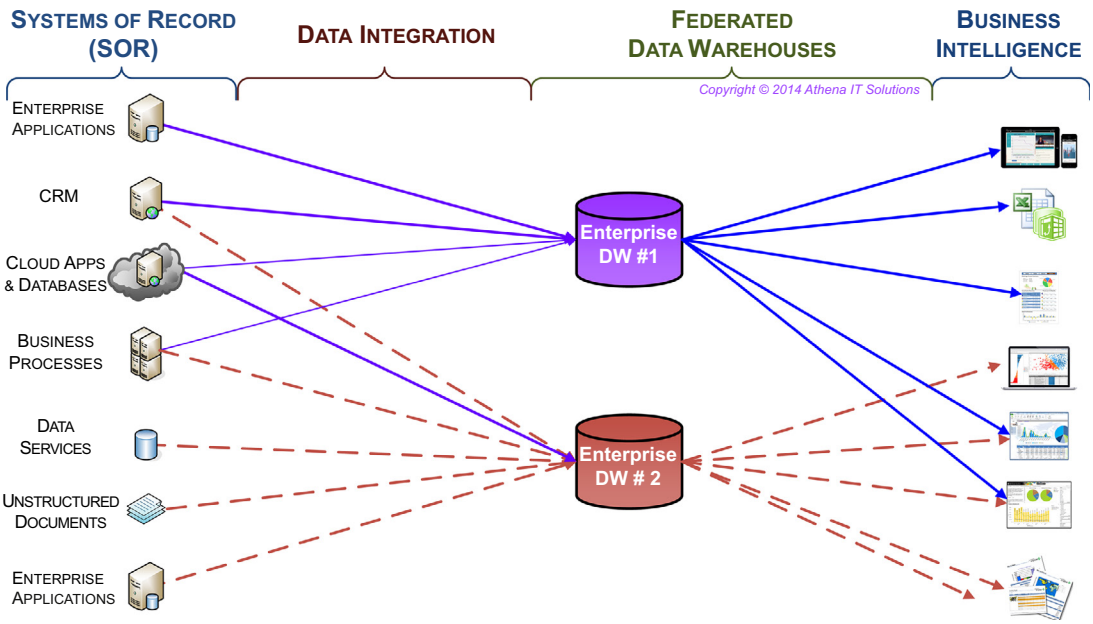

FIGURE 6.5

Federated DWs.

The federated DW suffers the same limitations as any of the other data architectures that do not use data marts to enable BI. These include performance issues, lack of DI, and no reliable source of historical data.

BI ACCIDENTAL ARCHITECTURE

The reaction to the independent data marts was to create DWs or operational data stores and drop them into an architectural smorgasbord, typically without redesigning the data marts. As depicted in Figure 6.6, it was common for an enterprise to have a patchwork of DWs, ODSs, and data marts along with multiple databases, ETL tools, and BI products. The idea of the accidental architecture is explained in detail in Chapter 5.

The current reality of many enterprise data landscapes is data silos that include overlapping and redundant DWs, ODSs, independent and dependent data marts, OLAP cubes, other reporting databases, and data shadow systems. A typical scenario could include the following:

- Sales and Finance have each built their own DWs independently without synchronizing data or transformations.
- The enterprise builds one or more ODS to handle near real-time updates.
- Other departments build multiple data marts, cubes, and spreadsheets for their specific business needs, and these systems have overlapping and sometimes conflicting data.

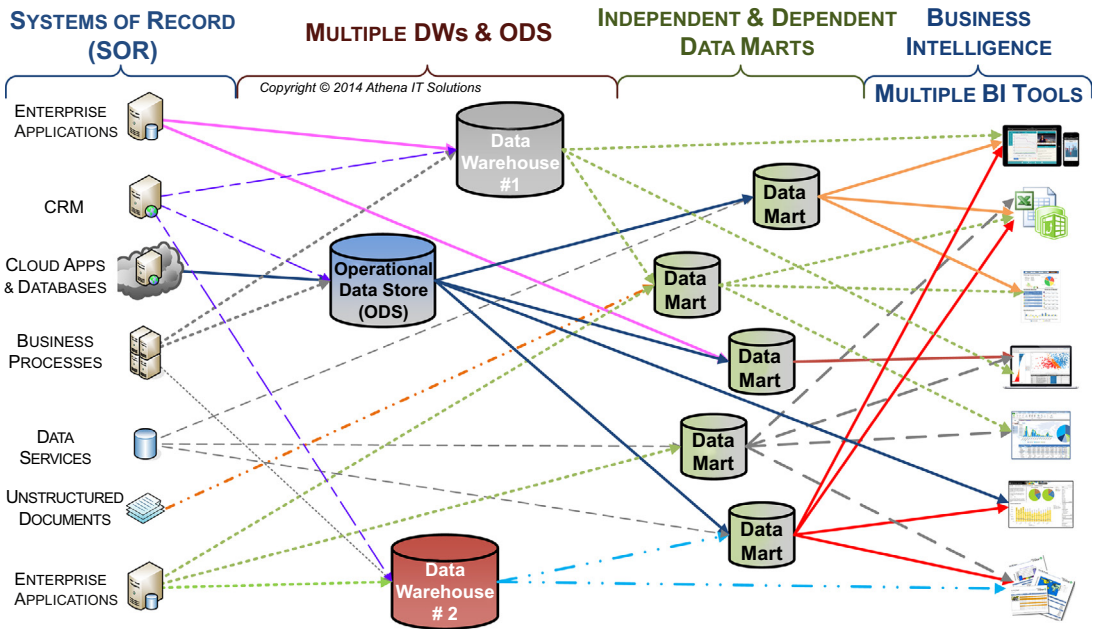

FIGURE 6.6

Multiple built BI silos & multiple BI tools.

Each of these BI data silos addressed a specific need, but looked at holistically they created their own silo problems. Despite the best intentions of each of the project-based BI applications, they were centered on the parochial interests of a specific business group and there was no overarching data architecture. It is inevitable that the solutions end up producing more harm than good when examined from an enterprise perspective.

HUB-AND-SPOKE

An answer to the data mart problem was needed. Two competing architectural approaches emerged: a data bus with conformed data marts versus a hub-and-spoke with a data warehouse feeding data marts, as shown in Figure 6.7. This set up the classic Kimball vs. Inmon (CIF) debate.

The conformed dimensions approach tied disparate data marts into a logical "single version of the truth" without the need for a DW, which appealed to those who still had a bias against DWs. The ETL layer staged all the data from the source systems and then directly parsed the data to the data marts. It was intellectually very appealing, but there were crucial stumbling blocks to implementing this approach:

- Only a small percentage of companies had the discipline to design and implement the required data integration processes.
- Inconsistency and data quality needed overly complex real-time DI that was beyond the ETL tools' capabilities at that time.

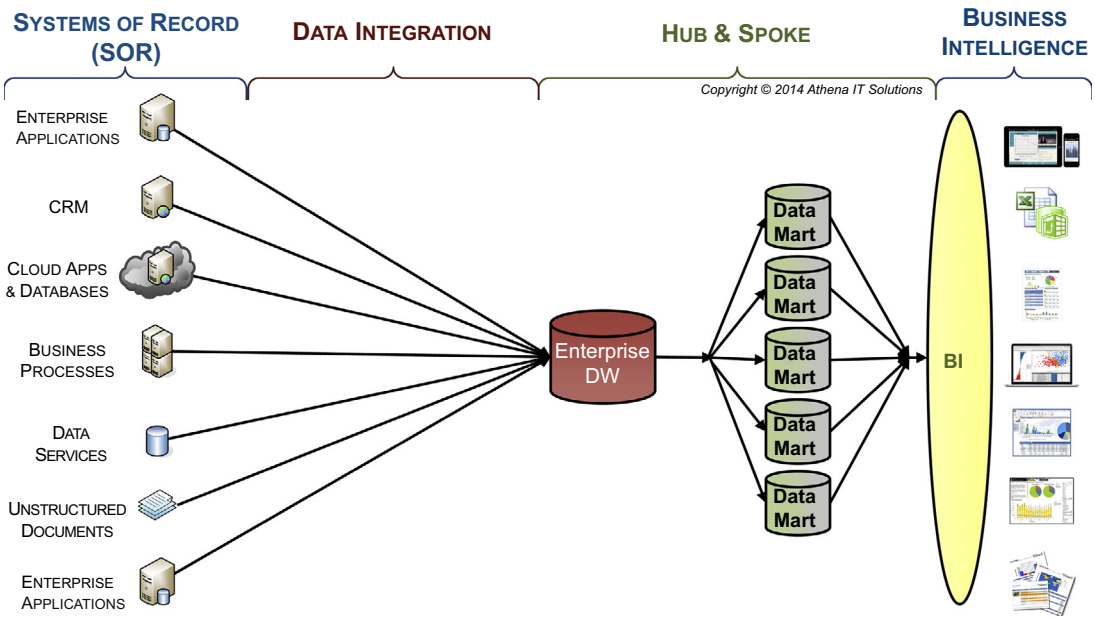

FIGURE 6.7

Hub-and-Spoke with one BI platform.

- In the real world, the timing of transactional data and the updates of its associated reference data were often out of sync, preventing the on-the-fly validation and integration of data from multiple source system to multiple data marts.

The hub-and-spoke approach became more popular, judging by the number of companies that adopted it; it ultimately became part of the architectural best practices. Successful hub-and-spoke implementations borrowed heavily from the conformed dimension approach by using its data modeling and integration techniques. The key difference was that the hub-and-spoke built a physical DW (data hub) rather than trying to achieve a virtual hub. That virtual hub was simple to draw but difficult to achieve in real-world situations.

Despite the adoption of the hub-and-spoke approach to unify data, however, the data silo genie was already out of the bottle. By the time companies adopted hub-and-spoke, they had already built countless ODS systems, DWs, data marts, online analytical processing (OLAP) cubes and data shadow systems. They didn't have the time or resources to bring these databases into the hub-and-spoke fold.

Although the hub-and-spoke enables the 5C's of information (consistent, clean, comprehensive, conformed, and current), some enterprises delegate BI application development and tool selection to departmental groups. This typically results in the scenario shown in Figure 6.8, multiple BI silos not only of different BI tools, but more damaging redundant and inconsistent reporting. Although the cause is organizational, it is important to consider this possibility even when an enterprise has achieved the 5C's. Refer to Chapter 19 on organizational best practice to avoid this scenario.

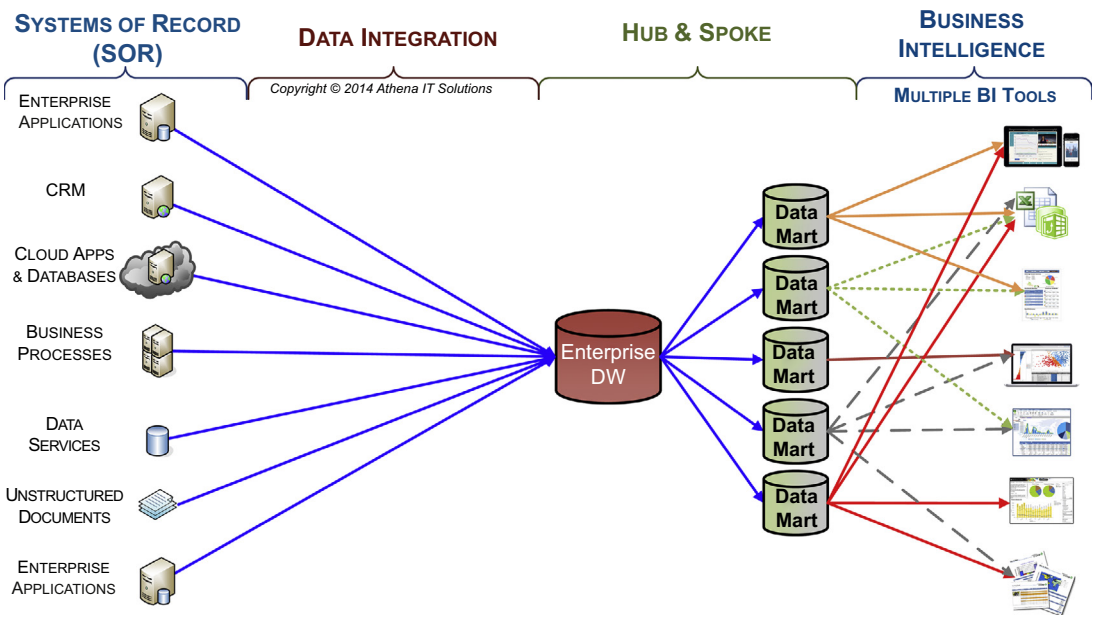

FIGURE 6.8

Hub and spoke with multiple BI tools.

DATA ARCHITECTURAL CHOICES

Various different data architectures have been used to implement enterprise BI solutions throughout the years. These architectures have evolved and often blended together, reducing many of the differences between them. Best practices have emerged, and, from a high-level perspective, include the recommended DI workflow and supporting data stores. From a lower level of detail, the best practices include the recommended DI processes, analytical processes, and the data schemas needed to support them.

Even with a long list of best practices, there still is much confusion and misleading concepts in the industry. The primary reasons for this confusion include the following:

- **Too much emphasis on technology**—there is still too much of an orientation toward technical architectures and products rather than information and data architectures. The reliance on technical wizardry may appear to work in the short run; sometimes "horsepower" can temporarily overcome poor design. But in the long run, without a fundamental data information architecture, the solutions tends to create costly application and data silos that are then replaced by the next round of overhyped technology.
- **Misunderstanding**—Without understanding the underlying data architecture best practices, such as data models and process workflows, designers who think they're building a particular architecture may actually build a system that is drastically different than what they thought they were building. This misapplication of best practices results in inferior solutions and gives them an undeserved bad reputation.

- **Old biases die hard**—initial reputations can be difficult to overcome, even for people who weren't directly affected. The underlying infrastructure (hardware, storage, networks, databases, interoperability, etc.), when much of the BI, DW, and DI concepts were initially introduced, was vastly inferior in comparison with today's standards. My iPhone has more memory, storage, processing power, and interoperability than the first DW environment I helped implement in the 1980s.

DATA CATEGORIES

There are three categories of data management processing based on business objectives and use cases:

- Data capture
- Data integration
- Information analysis (BI and analytics)

Regardless of advances in technology and products, these fundamental data management categories form the basis of an enterprise's data architecture and its information management. In the BI domain, the data architecture needs to address DI and information analysis, while the data capture category is managed by operational systems.

Since the beginning of DW, there have been significant technological advances as people expanded interoperability capabilities and learned from their mistakes and successes. But through all those changes and advances, the following fundamental concepts still need to be incorporated into designing data architectures:

- **There's a significant difference between data capture and information consumption in regard to their business uses and functional requirements**. Data capture is about creating, updating, or deleting data about a business event, transaction, or description of attributes relating to entities involved in those events or transactions. Information consumption is about analyzing those events, transactions, or descriptions in a context that is not material to it or may not even apply at the time. In addition, information consumption looks at data that is related to the business relationships and transformations that is not applicable at the point of capture.
- **Data captured at a point in time is a historical record that does not change**. (Although it may be altered because of error, other changes that are the result of subsequent events or transactions do not change the underlying historical event.) But with information consumption, data does change—it evolves as the context of business data relationships changes. These changes can include product hierarchies, sales territories, contractual relationships, organizational structures, and customer attributes.
- **DI keeps getting more complex** not because of volume or velocity, as that can generally be handled by technology advances, but rather because of the variety and diversity of sources and business interpretation. Data silos keep propagating because of politics and parochial interests, requiring ever-increasing DI.

SELECTING A DATA ARCHITECTURE

The major competing data architectures are:

- EDW
- Independent data marts

- Enterprise data bus architecture—Ralph Kimball [1]
- Hub-and-spoke, corporate information factory (CIF)—Bill Inmon [2]
- Analytical data architecture (ADA)

The history lesson earlier in this chapter covered the EDW and independent data marts. Although they provide reporting, they are not likely to provide the five information C's (consistent, clean, comprehensive, conformed, and current). Each of these architectures is too focused on only one of the two data management categories:

- **EDW-only architecture addresses DI**, but makes it extremely difficult for business people to use it for information analysis. Historically, when IT was responsible for creating all reports, this was acceptable because they could navigate the labyrinth of a DW schema. But now that we have self-service BI, it's no longer viable to have data discovery and data visualization tools accessing the EDW.
- **Independent data marts address information analysis** but at the expense of not integrating data and creating silos. In the short term, this approach may create reporting applications, but it lessens business productivity with the time lost on reconciliation and debating the numbers; costs more because of overlapping and redundant activities; and poses too much business risk due to inconsistent, inaccurate and incomplete data. The resulting data silos create more problems than they solve and increase the need for further DI.

The two divergent BI camps emerged in the 1990s to address the limitations of the EDW-only and independent data mart architectures:

- Bill Inmon's CIF, as shown in Figure 6.9.
- Ralph Kimball's enterprise data bus architecture, as shown in Figure 6.10.

> See the book website www.biguidebook.com for more on the Bill Inmon's CIF and Ralph Kimball's enterprise data bus architectures. Figures 6.9 and 6.10 are my simplification of these architectures and represent the "evolved," rather than the initial, versions.

Table 6.1 lists the key differences between Inmon's CIF architecture and Kimball's enterprise data bus architecture.

The fundamental differences of their approaches are:

- CIF started out with the EDW as the core of the architecture, while Kimball's architecture focused on the BI (and business-oriented) data marts.
- CIF is a top-down design creating an enterprise-wide data model from the source systems to design the EDW, while Kimball uses a bottom-up business requirements-driven approach to design schemas for the data marts. The key assumptions when the respective architectures were introduced were:
  - Inmon's architecture created an EDW organized by subject areas containing detailed data and departmental databases. These were organized by their parochial interests, typically with summarized or aggregated data.
  - Kimball's architecture assumed the ETL system could perform the 5C's (consistent, clean, comprehensive, conformed, and current) either on-the-fly or using a staging area and thus making the need for an EDW superfluous.

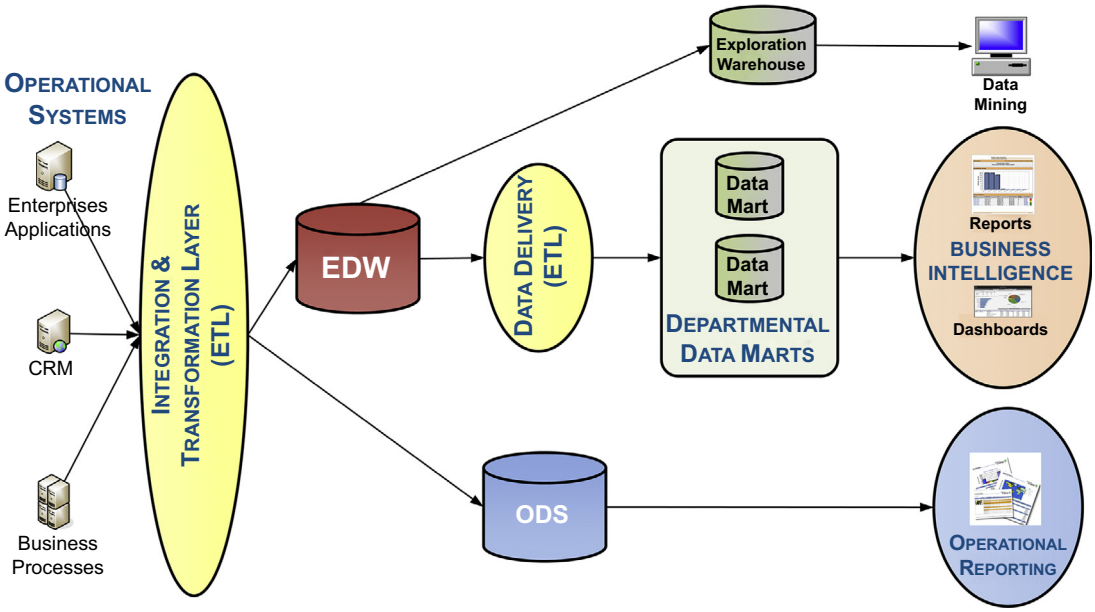

FIGURE 6.9

Inmon's CIF architecture.

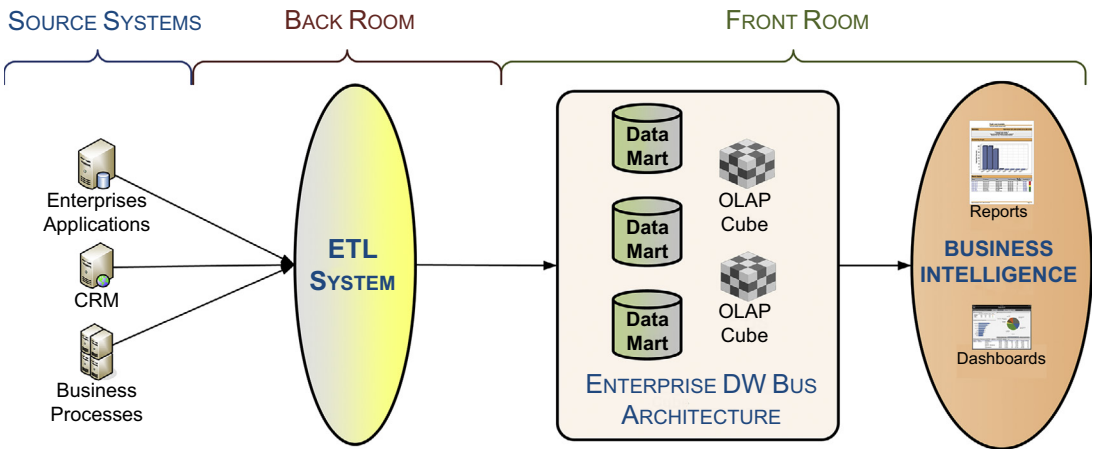

FIGURE 6.10

Kimball's enterprise data bus architecture.

Table 6.1 Comparison of Inmon vs Kimball Architecture		
Concepts	**Inmon's CIF**	**Kimball's Data Bus**
Key building block	EDW	Data marts
Data management process focus	Data integration	Business intelligence
Scope	Enterprise-wide	Business processes
Key data modeling technique	Normalization for EDW only; marts can be any design	Dimensional models
Data warehouse	Yes	Unnecessary
Data marts	Summarized from DW	Yes

- Inmon believed that the EDW needed to be normalized, i.e., implement a third normal form (3NF) schema, while Kimball used dimensional models for the data marts and felt the 3NF requirement of the CIF was unnecessary and too costly to create.

Both architectures have evolved to where the use of an EDW and data marts, i.e., hub-and-spoke architecture, is not only acceptable but recognized as the most pragmatic approach. There is still debate, however, about 3NF versus dimensional models. Both camps have taken a more holistic view regarding data management processes, but they still retain their initial biases to a particular data management category.

Despite the history, the hub-and-spoke architecture with an EDW feeding data marts was created in the mid- to late 1980s. Industry best practices have finally evolved to embrace the hub-and-spoke architecture of the earliest adopters.

THE SAME BUT DIFFERENT

Even though hub-and-spoke is considered the industry best practice, the results and costs can be vastly different even in comparable situations. This is because of significant variations in the underlying design details, especially in the data models and processing workflow. Many of these variations result in higher costs, slower response to business needs, and, ultimately, the BI system not being used or regarded as the system of analytics. This encourages the creation of more data silos either in the form of independent data marts or data shadow systems, resulting in higher costs and a loss in productivity, and potentially creating business risks due to inconsistent or incomplete information. These varying design details may cause enterprises to have an unfavorable view of what is otherwise a good industry best practice.

The most common mistakes made by enterprises implementing the hub-and-spoke architecture are:

- Designing the EDW using a "standard" 3NF schema (Chapter 8 explains 3NF) rather than a hybrid dimensional-normalized model
- Allowing the business BI consumers to perform self-service BI from EDW
- Designing a separate data mart or BI data structure for each dashboard or report
- Designing data marts to support many business processes or entire business groups
- Thinking that a BI product eliminates the need to design a logical dimensional data model

The most common mistakes made by enterprises implementing Kimball's architecture are:

- Designing each data mart to be so parochial to a department's terminology and business definitions that conformed dimensions and facts across data marts are impossible
- Designing a separate data mart for each dashboard or report
- Abandoning dimensional models by denormalizing data to ease report development or improve query performance
- Failing to implement slowly changing dimensions (SCD) to track historical changes in data (this often happens because departments did not initially think they needed it for their reporting)
- Thinking that a BI product eliminates the need to design a logical dimensional data model

In the beginning of DW, many enterprises built or bought transactional systems that used a variety of database and file structures that did not use a 3NF schema. With the source systems having a variety of schemas, it was a best practice to build the EDW using a 3NF schema, as that was the only location in the enterprise that the data was normalized. (The benefits of normalization are presented in Chapter 10.) Bill Inmon's CIF architecture was conceived in this era and thus its normalization requirement for the EDW was an industry best practice.

As the 1990s progressed, the following major trends in the scope and structure of operational systems affected the design of DWs:

- Enterprises replaced legacy operational systems that were often custom-built with enterprise resource and planning (ERP) systems that they bought from IT vendors.
- IT vendors built their ERP systems using relational databases with 3NF schemas.
- The ERP systems consolidated many previously silo'd operational systems into an application that had a more enterprise-wide scope.

These trends eventually shifted the responsibility of designing a 3NF data model from the EDW to source systems, such as ERP applications. During this transition, best practices emerged for data integration, BI and advanced dimensional modeling that enabled the standard 3NF being used in EDW to evolve to a hybrid dimensional-normalized data model (explained later in this chapter.)

During the early phases of the rise of ERP systems, IT vendors sold the idea of one-stop shopping for operational systems with their own dedicated reporting. This resulted in warring IT camps between the ERP and BI implementations, with each side feeling they were developing the "single version of the truth" and competing data silos. Eventually, IT vendors came to the realization that the ERP systems did not, and likely would never, contain all the data than an enterprise needed for reporting. With this realization, the IT vendors that sold ERP systems embraced DW and sold their own BI solutions. Unfortunately, IT groups designing DWs have often continued to use 3NF schemas even though it is no longer necessary or the best practice because 3NF is what they are familiar with.

ANALYTICAL DATA ARCHITECTURE (ADA)

I recommend the implementation of the ADA depicted in Figure 6.11. This architecture represents the evolution of best practices for BI, DI, and DW with a framework that can leverage current and future technologies in its implementation. Keep in mind that with a high-level diagram like this many of the underlying design best practices and exceptions are not visible.

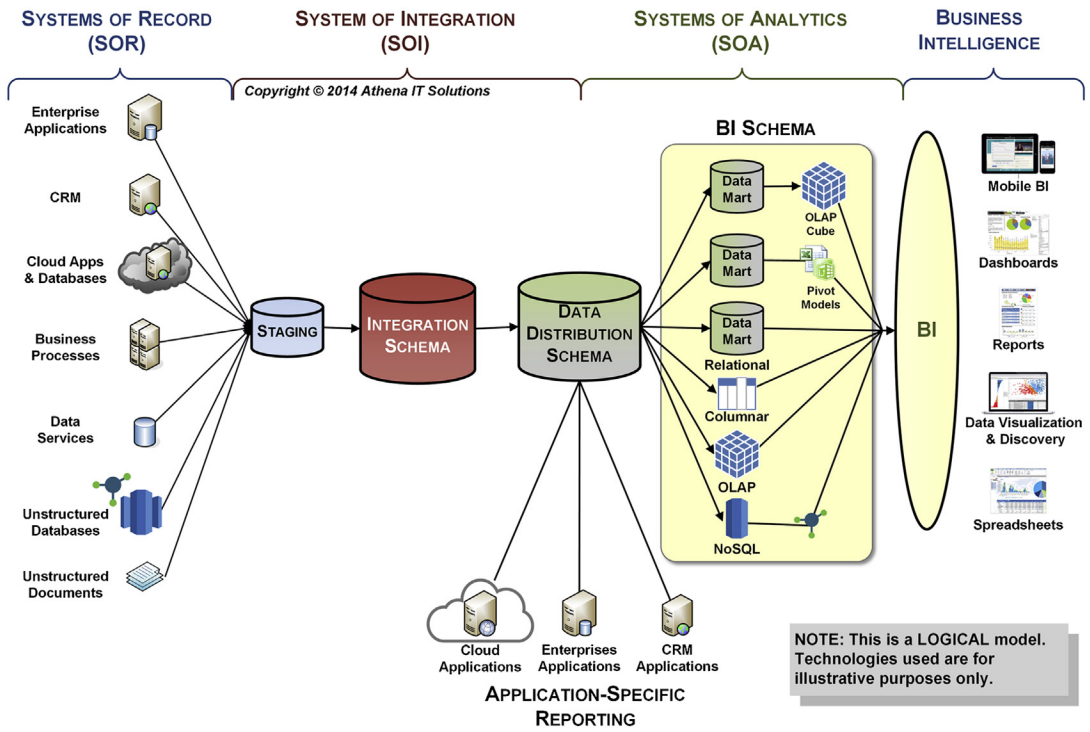

FIGURE 6.11

Analytical data architecture (ADA).

There are three layers of data that enable BI:

1. **Systems of record (SOR)**—source systems where data is created, updated and deleted. This layer is the source of data for systems of integration (SOI) layer but is managed outside the ADA.
2. **Systems of integration (SOI)**—data structures used to integrate data to enable the 5C's of information: consistent, clean, comprehensive, conformed and current.
3. **Systems of analytics (SOA)**—data structures used by BI application to enable analytics

Several aspects of the underlying design details are explained in other portions of this book:

- Data integration workflow—the next section.
- Data schemas—Chapter 10.
- Technologies—Chapter 7.

Enhanced Hub-and-Spoke Model

The SOI replaces a lone physical EDW with SOI feeding the SOA. The goal of the SOI is to integrate data to create comprehensive, current, clean, and consistent information for the enterprise. The goal of the SOA is to enable BI by providing business-oriented information for analysis.

The SOI logically contains an integration schema and staging area. The integration schema will be a physical data store, while the staging area may be persistent or transient based on use cases. The latter is the case when the DI processes do not need persistent data stores for acquiring or integrating data.

The SOA contains the logical data distribution schema that potentially feeds data marts that, in turn, may feed sub-marts such as OLAP cubes or pivot tables. The SOA is a source of BI data either by BI tools accessing the data or by operational applications receiving distributed data from the data distribution schema. The latter would prevent operational applications from redundantly sourcing data from other applications when the EDW has already done so.

Enterprise Data Warehose (EDW)

The EDW is split into two schemas with different data objectives. The two schemas are:

- EDW integration schema—its purpose is to enable the enterprise DI processes.
- EDW data distribution schema—its purpose is to distribute the data downstream to the data marts so BI consumers can analyze the data or do application-specific reporting and advanced analytics that support the business.

Although the EDW is the source of the enterprise's integrated 5 C's data, it is pragmatic to recognize that not all data needed for analytics will be available in the EDW. The intent of this chapter is to discuss how the integrated data that does flow though the EDW will be designed and processed. Using data both from the EDW environment and directly from data sources is discussed in Chapter 15. The Information (Chapter 5) and Technology (Chapter 7) Architectures discussed in this book also incorporate analytics accessing data from the EDW and other data sources.

Data Marts

The EDW data distribution model represents the enterprise dimensional model that feeds both data marts and application-specific reporting. Data marts are designed to support business processes or business groups. Sub-data marts are created to support different types of analysis performed within a business process or a business group. These sub-data marts are fed from data marts and derived by applying additional filters, business rules, and transformations that are specific to a particular business line of analysis.

The dimensional model and data workflow depicted in the ADA are logical rather than physical. The logical dimensional model may be implemented in a variety of database technologies and the data workflow may be built in one or more databases (and database technologies). Both are discussed further in this chapter.

Data Schemas

The EDW staging area, if it is a persistent data store, may use a variety of schemas such as relational, flat file, or XML, depending on its use case. There may be a variety of EDW staging area data stores to support acquiring data from a similar variety of different source systems.

The EDW integration schema uses a hybrid dimensional-normalized data model. This schema needs to be built from a 3NF or normalized data model representing the entities, attributes, and relationships from the source systems that are relevant to BI with model extensions (see below). If the

underlying source systems do not have supporting 3NF data models, then create those models from the extracted data. There are several key purposes for using a hybrid model:

- Leverage 3NF design principles to promote normalization.
- Leverage advanced dimensional modeling constructs to enable tracking dimensional model changes and eliminating hierarchical structure changes.
- Implement data lineage, auditing, cleansing, and tracking in data integration and BI workflow.

Although not depicted in the diagram, if an ODS is used in the ADA, its schema will be based on its intended use and the database technology it is implemented with. The ODS schema could range from a 3NF in a relational database to a schema-less NoSQL database.

The EDW data distribution schema, data marts, OLAP cubes, and any other SOA data store use a logical dimensional model. This becomes a physical dimensional model if a relational database is used, however, different technologies may physically deploy other schemas such as multidimensional OLAP cubes, columnar databases, in-memory databases, and semi-structured or unstructured databases. See Table 6.2.

Table 6.2 BI Data Stores and Data Model

Data Store	Data Model
Staging	Same as source system
ODS	3NF or alternate schema based on its use
EDW: Integration schema	3NF or hybrid dimensional-normalized
EDW: Distribution schema	Hybrid dimensional-normalized
Data mart (relational)	Dimensional
OLAP cube, columnar, in-memory and other non-relational BI data stores	Logically dimensional
MDM (for source system applications)	3NF
MDM (for BI only)	Hybrid dimensional-normalized

Understanding the Hybrid Dimensional-Normalized Model

I have been able to persuade both normalization and dimensional zealots to design, develop, and deploy EDWs with this hybrid dimensional-normalized model. Rather than compromising their respective viewpoints, this hybrid model leverages the best of both to integrate data in an EDW.

From a 3NF modeler's viewpoint, this model leverages a 3NF data model while also:

- Grafting on several advanced dimensional modeling constructs (below).
- Extending the data model to track data integration and analytical processes.
- Consolidating 3NF entities when specific entity relationships and other entities are not relevant to BI.

From a dimensional modeler's viewpoint, this hybrid model leverages a standard dimensional data model while also:

- Using advanced dimensional modeling constructs (below).
- Extending the data model to track data integration and analytical processes.

- Normalizing dimensions using outriggers when dimension tables have groups of attributes at different grains or are updated at much different frequencies (referred to as rapidly changing dimensions).
- Avoiding the use of summarizations or aggregations unless required by a business transformation.

The advanced dimensional constructs include:

- Implementing a date dimension and add date dimension foreign keys for each of the date attributes in fact tables while retaining the original data attributes.
- Creating slowly changing dimension (SCD) data models to track historical changes to dimensional values. This leverages documented processes that are built into data integration processes.
- Using bridge tables to model hierarchies rather than using separate tables for each hierarchy level. This avoids changing the physical number of entities in the model, as the number of hierarchy levels inevitably changes.
- Using bridge tables to many-to-many valued relationships.
- Creating surrogate keys, independent of source systems' natural keys, for all tables, not just tables that are considered dimensions, to facilitate independence from source systems.
- Logically classifying entities as dimensions, facts, bridges, or other types of tables. This assists in following the best practices for data integration and conversion to dimensional models for BI.

The extensions to the data model, as discussed in Chapters 10–12, include such attributes as data integration process identifiers, time stamps for when data was created or modified, source system identifiers, error handling, data cleansing, security and privacy roles, and other BI/DW life cycle-specific attributes.

Source systems, especially ERP, SCM, and CRM applications, may have 3NF data models that contain thousands or tens of thousands of entities and relationships. A significant portion of those entities and relationships are needed by application-, transaction- or business-event-processing, but are not relevant to BI. This creates a subset of entities, attributes, and relationships relevant for BI that can be consolidated. Although this consolidation might be considered denormalizing the 3NF since entities, attributes, and relationships have been eliminated, the integrity of the 3NF model is maintained.

BI Sources

In the early days of DW there was a presumed edict that all reporting and analysis, i.e., BI, would be sourced solely from the DW environment. There are two primary reasons why this is not practical:

1. Is highly unlikely that the BI environment will source all the data that any business person would need to analyze at any time.
2. Much of an enterprise's operational reporting and analysis can be accomplished by directly accessing the operational systems, provided that these systems have all the data needed for analysis.

When either of these two conditions occurs, the BI tools will need to directly access the operational source systems. As discussed in Chapter 7, tools such as data virtualization make that access easier and more transparent to the business person. The enterprise IT group must handle both types of analysis in the data management framework.

Physical Data Store Combinations

EDW data distribution schema, data marts, OLAP cubes, and any other SOA data stores are logical, not physical, and based on the data use case, one or more of these data stores may not need to be made a persistent physical data store.

Each of the data stores may actually be split into federated entities. For example, the EDW may be split into federated DWs based on such criteria as geographic regions, business functions, and business organizational entities or to support structured versus non-structured data.

On the other end of the spectrum, the entire set of data stores may be implemented on a single database platform, with each data store being represented as a schema within that database.

DATA INTEGRATION WORKFLOW

The data architecture enables the transformation of data into consistent, conformed, comprehensive, clean, and current information for business analysis and decision-making. The diagram in Figure 6.12 illustrates the flow from data creation in the SOR through information consumption by business people

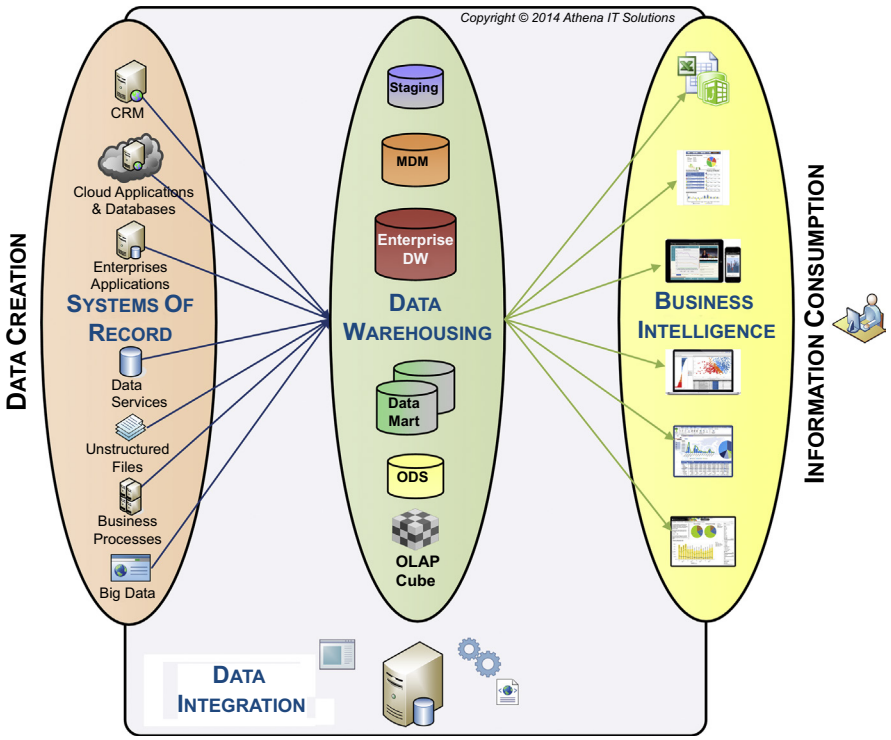

FIGURE 6.12

Data integration workflow.

using BI. Data is transformed into information by DI processes (depicted in the example's data warehousing section) that:

- Gather data
- Integrate and transform data using business rules and technical conversions
- Store it in various data stores
- Make it available to business users with BI tools

An analogy to this process is the material flow of a consumer product from the supplier (SOR), to factory (DI), to warehouses (DW), and finally to retail stores (BI applications), where consumers buy it (business analysis).

The following sections progressively explain in more detail the various data stores that may be deployed within the ADA.

DATA INTEGRATION WORKFLOW—HUB-AND-SPOKE

The initial starting point for discussion of the DI workflow is a classic hub-and-spoke architecture, with an EDW gathering data from SORs and then feeding data marts that are accessed by business people using BI, as shown in Figure 6.13. The central premise of this architecture is that the heavy-duty DI activities, i.e., data preparation, are completed as the data moves from source systems into the DW where it becomes the SOI. Once done, the data is propagated, i.e., data franchised, from the DW into the data marts. Data franchising packages the data via filtering, aggregation and transformations into SOA to be consumed by BI.

This diagram begins the discussion on the functionality and supporting data stores for the DI workflow, while the following sections will decompose the SOI and SOA into more detail.

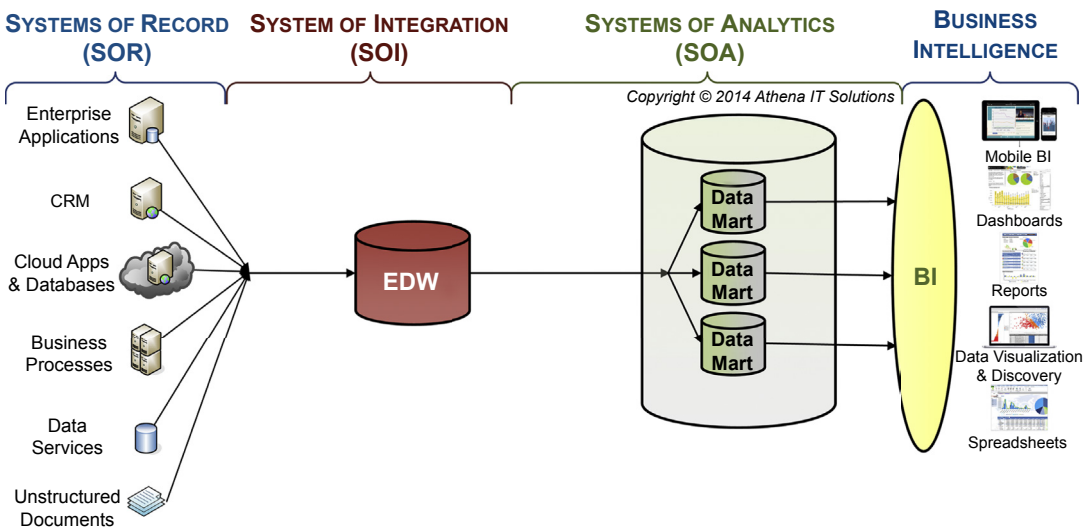

FIGURE 6.13

DI workflow—hub-and-spoke.

DATA WORKFLOW OF THE SYSTEM OF INTEGRATION (SOI)

The SOI has three primary functions:

1. Gather data from the SORs—stage
2. Perform data preparation—integrate
3. Serve as a data hub for the SOA—distribute

Figure 6.14 depicts the breakdown of the SOI from the EDW by itself into three data stores that support its primary functions. Table 6.3 pairs function and data store in tabular form. These data stores are logical entities that may become either a physical, persistent or a virtual, transient entity. At a minimum, the EDW will be a physical entity, whereas the staging area may be transient.

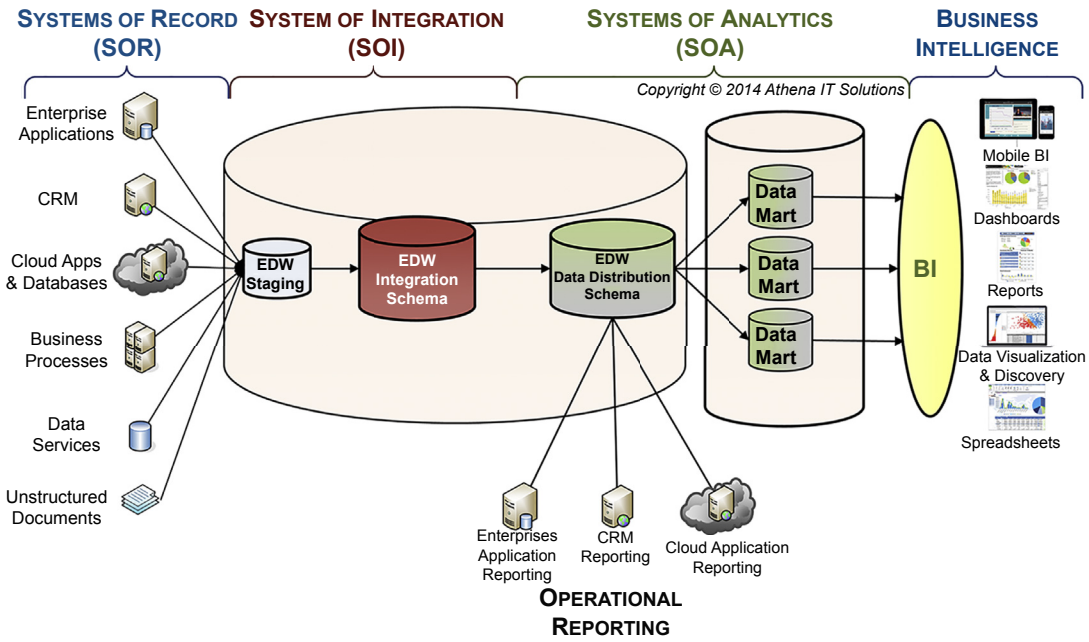

FIGURE 6.14

DI workflow—SOI data stores.

Table 6.3 SOI Functions & Data Stores

Function	Data Store
Stage	Staging area (also called landing area)
Integrate	EDW integration schema
Distribute	EDW distribution schema

Staging (or Landing) Area

With today's database connectivity and API access to most of the widely-used ERP applications, it is relatively easy to directly connect, query, and extract data. In addition, many DI tools enable the use of memory and parallel processing when gathering and integrating data from multiple sources. It would seem, therefore, that physically staging data would no longer be necessary, but for a variety of reasons it is often a best practice to maintain a staging area feeding the EDW.

There might be a requirement to stage data during other DI or access processes throughout the DI workflow; however, those are typically handled by transient data stores or processed in memory.

There are several reasons to implement a physical staging area: audits; recover and restart; referential integrity; data quality; recycling rows and error handling; and enterprise standard or source system requirement.

- **Audit**—Auditing enables you to provide detailed documentation of the workflow of data from source systems through information consumption. This provides several benefits such as:
  - Impact analysis to determine the effect of changes in source system data on the BI solution
  - "Where used" search capability so that IT and business people can examine where the data they are using originated
  - System documentation for maintenance and enhancement.
  - Regulatory and legal requirements to provide auditing

A DATA AUDITABILITY STORY

A former colleague of mine experienced first-hand one of the pitfalls of not knowing the lineage of a client's data. The project was to replace a spreadsheet-based budgeting, planning, and forecasting system (data shadow system) with a performance management solution at a multi-billion dollar company. He spent 10 hours in a client meeting—one of those painful marathon meetings—where they were discussing the design of the new system and poring over the data in the client's spreadsheets to understand how the data was transformed and manipulated.

The meeting dragged on through lunch, and then dinner. Finally, a senior manager from the client said "Hey, wait a minute. This isn't even the right set of spreadsheets!" Everything that the dozen people at the meeting—half from the client's finance staff and the other half highly paid consultants—had just done over the past 10 hours was a total waste of time.

The sad truth is that my colleagues and I see many clients who honestly have no idea how their data got into their reports or Microsoft PowerPoint slides. They may know what enterprise application the data originated in and can get their IT staff to document how the data was loaded in their DW, but then the trail goes cold.

Under the circumstances, the staging area will follow enterprise standards relating to backups and archiving. This further ensures you can audit historical extracts or perform disaster recovery processing.

- **Recovery and restart**—Although it is generally possible to re-query the source systems if there is a failure when loading the EDW, sometimes it is preferable to maintain a copy of the data extracted from the source systems in the staging area to be used when recovery and restart is necessary. The criteria to select this approach are:
  - When loading a periodic snapshot, i.e., a data snapshot, at a point in time
  - When a stage copy would avoid having to redo lengthy extraction times
  - When there are limited times when the source systems are available for extraction
  - To enable debugging loading failures

- **Referential integrity**—It is a BI best practice to enforce referential integrity during DI processing rather than using database constraints. Enforcing referential integrity requires that the attribute that the foreign key is related to, typically a dimensional attribute, is available for validation at the point in time that the foreign key needs to be checked. Dimensional reference data used in this validation is often managed by separate business processes other than transactional or operational systems. Under those conditions it is common to have early-arriving facts or late-arriving dimensions, necessitating the need to stage data to make it possible to validate referential integrity in real time.
- **Data quality**—The bulk of the data quality and cleansing processes occur in data preparation (see Chapter 12). There are conditions, however, when it is better to perform this during the initial loading process into the staging area. The general rule of thumb in quality processing is to correct errors as soon as possible, as they get more expensive and difficult to correct the longer it takes. It is best to use the staging area for data quality processes in those cases when waiting until later would make it much more complex or potentially even impossible to correct.
- **Recycling rows and error handling**—As mentioned in the referential integrity bullet, there may be circumstances where DI workflow encounters early-arriving facts or late-arriving dimensions. When these conditions occur, the referential integrity checks would flag an error in processing and prevent it from moving the data forward in the workflow. At this point, DI processes need to know how to proceed. The most extreme option would be to stop processing all the data in the load at the time of a referential integrity error. Although this is a viable option in certain circumstances, in general the offending rows are rejected and processing continues. The next decision is what to do with the rejected rows. The options are to just ignore them as if nothing happened (the least likely alternative), load the offending data with an indicator to flag the error condition, or suspend the data and attempt to reprocess it on subsequent data loads. This last condition is a typical practice that requires the suspended data to be stored in the staging area.

  The process used to handle referential integrity errors should be defined in the business requirements that the BI team obtains while working with the business. Typically this level of detail is not obtained when requirements are initially documented, so the BI team will need to revisit this with the business to make sure these processing requirements follow business rules.
- **Enterprise standard or source system requirement**—Often a staging area is required due to an enterprise IT standard or source system requirement. The SOR, for example, may not be available for direct access, or for performance reasons, no direct access is allowed by the BI team's DI processes. Under these conditions the SOR application team will provide required extracts, typically in files that are placed in the staging or landing area.

Splitting the EDW

The EDW has two distinct functions: integrating the data and then distributing it. Although one schema could support both of these functions, is much better to divide the EDW into two schemas, each supporting one function.

The purpose of the EDW integration schema is to enable the data preparation integration processes and to manage historical data. The hybrid dimensional-normalized model is the most effective at performing these processes:

- Tracking the inevitable changes in business and data relationships
- Minimizing or possibly eliminating the need for data structures with routine business changes.

The EDW integration schema's hybrid dimensional-normalized model is not as complex in terms of tables and data relationships as a normalized model, but it is much more complex than the data models that are required in the SOA data stores. The data franchising process takes data from the EDW and propagates it into each of the data marts in the SOA. Besides the business transformations that the data franchising processes perform specific to each data mart, every data franchising process needs to perform data restructuring between the EDW and the data mart schemas. The restructuring converts the hybrid dimensional-normalized model into a standard dimensional model by denormalizing or flattening the structures. In addition, most of the data marts will not require the structures used to track changes in hierarchies or other historical attributes. The data restructuring process is redundant across each of the data franchising processes and needs to be re-created every time a new data mart is created.

It is best practice to perform the data restructuring once and reposition the data franchising processes to concentrate on the business-specific transformations from the EDW to individual data marts. Creating an EDW data distribution schema with the type of dimensional model that is used in data marts enables the data restructuring to execute once when it gets loaded from the EDW integration schema and then be used many times as the data franchising processes load data marts.

In addition to being the source of data for the SOA, the EDW data distribution schema also serves as a data hub to operational reporting.

DATA WORKFLOW OF THE SYSTEM OF ANALYTICS (SOA)

The primary goal of the SOA is to enable business people using BI to analyze information, make more informed decisions, and potentially improve the operations or profitability of the enterprise. This is accomplished by taking the data from the SOI and performing data franchising processes to package data to support analysis for specific business processes or business functions.

The classic diagram illustrated in Figure 6.13 depicts an EDW feeding a single tier of data marts that are then accessed by BI. This is a good starting point to explain the need for data marts in the purpose of data franchising, but it is a too simplistic approach for most enterprises, whether they are Fortune 500 or small to medium business (SMB) enterprises.

As depicted in Figure 6.15, the SOA has multiple tiers of data marts. The diagram provides a logical depiction of many data stores, but there are several implementation choices available:

- One physical data store, such as a database housing all the logical data stores represented by separate schemas. This physical data store could also include the SOI data stores.
- Multiple physical data stores all using the same technology, such as a database.
- Multiple physical data stores using different combinations of technologies, such as relational database, columnar database, OLAP cube or an in-memory database managed by a BI tool, spreadsheet, or BI appliance.

Different physical schemas may be required by these technologies. But, regardless of the physical implementation, each of the SOA data stores should have a logical dimensional model that defines the schema in the context of business rules and relationships.

The first tier of the SOA is the EDW data distribution schema (enterprise data mart). (Note: when discussing why we split the EDW in Figure 6.14 we placed the EDW data distribution schema in the SOI, but its proper positioning in the analytics data architecture is the SOA.) As previously discussed, this is primarily used to improve the efficiency of schema transformations that are required when moving from an EDW integration schema to the data mart's dimensional models.

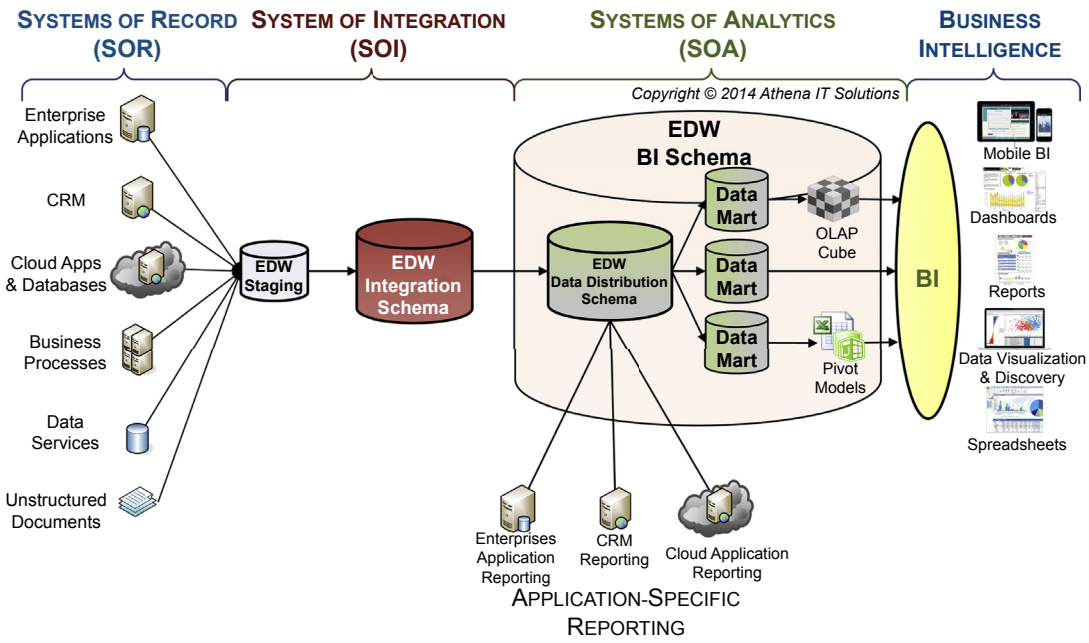

FIGURE 6.15

DI workflow—SOI & SOA.

In the second tier of the SOA are the classic data marts that are built by franchising data from the EDW data distribution schema (enterprise data mart) to support analysis for specific business processes or business functions. Subsequent tiers, i.e., sub-data marts, are created by applying additional data franchising processing that further refines the data for a particular analysis that a subset of business people using the original data mart need. This is a stepwise refinement process that continues to find new analytical uses for the data in the SOA.

The Need for the SOA

The stepwise refinement process of creating multiple tiers of data marts takes more time in the short run than constructing the classic single tier of data marts. This approach, however, is a very efficient method to prepare the data for reporting and analysis that business people perform in their jobs. Without these multiple tiers, every report and business analysis would need to repeatedly perform the same filters, aggregations, and transformations of data—work that could have been done once in a sub-data mart.

If you do not design and deploy multiple tiers in the SOA, then each BI application (e.g., reports, dashboards, data discovery tools, and data visualizations) will need to embed the data franchising process logic that would have been used to build the sub-data marts. This means that the BI applications are more complex to design, build, and maintain; queries are slower; and analysis is more difficult because business people need to understand much more complex data structures. The multi-tier data stores, on the other hand, are prebuilt for better understanding and easier access, and they are customized for their customers. They boost productivity for both IT and business because the logic to build the sub-data marts needs to be done only once.

By adopting a single tier of data marts, enterprises shift the burden of data franchising onto their BI tools and, ultimately, their business users. This has the following two significant adverse effects:

- **BI applications take more time to build, operate, and maintain.** You shift the cost of building a data mart once to building the same logic many times into a BI report or business user's analysis. From a total cost of ownership (TCO) perspective, this shift increases an enterprise's cost of providing the data for analysis and reduces the productivity of the business as it tries to analyze the data.
- **Creation of data shadow systems.** The second and more costly result of a single tier of data marts is shifting the data franchising processes from IT creating the multi-tier data stores to the business creating multiple data shadow systems. Chapter 16 describes these systems and their adverse impact on the enterprise. In particular, data shadow systems create data silos of inconsistent information, reversing the benefit of having a BI solution.

Benefits of the SOA

Although the classic single tier of data marts is simpler to implement in the short-term, the BI team should strive to explain and justify the more sophisticated and robust SOA architecture.

The benefits of SOA's multi-tier data stores are:

- **Enables self-service BI**—BI tools are continuously evolving to improve their ease of use and increase their business analytical capabilities with the goal of providing self-service BI. Regardless of BI tool capabilities, in order for self-service BI to occur, the data architecture needs to provide the 5C's of information (consistent, comprehensive, conformed, clean, and current). But an additional C is needed: context. Each data franchising process and SOA data store provides greater business context that improves understandability and business productivity, increasing the likelihood that self-service BI succeeds.
- **Increases business productivity**—If the enterprise does not create the tiered data stores with the data franchising processes, the burden of re-creating those processes will be on the shoulders of the business person who is performing analysis using BI tools. It means that every time that person goes into a data discovery, data visualization, or dashboard tool, she needs to re-create the filters, groupings, business transformations, and aggregations that could have been prebuilt in a data store.
- **Enables more efficient data processing**—By shifting the data franchise processing from the IT developer creating a BI application or the business person using a self-service BI tool to the SOA, overlapping or redundant data franchising processing is eliminated. This improves overall processing efficiency. Similar to an EDW that sources integrated data once for the enterprise so it can be used many times, the SOA formulates preconfigured data stores once so multiple business people can use them repeatedly, using many BI applications.
- **Lowers TCO and increases business return on investment (ROI)**—On the cost side of the equation, improved processing efficiency reduces the cost and support of additional infrastructure that will be needed without the SOA. In addition, improved business productivity should also reduce overall costs or at least shift activities back to business people's "real" jobs because they're no longer spending time processing data.

Self-service BI and more understandable business information will prove the business value of BI and, in turn, increase its ROI.

DATA WORKFLOW—RISE OF EDW AGAIN

The industry has adopted the hub-and-spoke architecture as a best practice, and the ADA is a more detailed refinement of that approach. Although Chapter 7 will review technologies in more detail, there is one trend that needs to be addressed in designing the SOA: the rise of the EDW.

The industry has shifted back and forth between a centralized and distributed deployment of a BI architecture. The first wave of BI was a distributed architecture, primarily because the technology firms that were the earliest adopters were using servers, at that time called minicomputers, for implementation. The second wave in the 1990s was spearheaded by using mainframes to implement a centralized DW. The third wave in the timeline reverted to the distributed architecture model with, initially, the use of client/server, and then evolving to use many distributed devices and services.

The industry is currently undergoing a fourth wave where, due to advances in databases, storage, network, and server capabilities, it is often more effective to deploy the hub-and-spoke all on the same database, BI appliance, or cloud platform (as depicted in Figure 6.16) rather than store the EDW (hub) and data marts (spokes) on separate servers and databases. Although this is often a very cost-effective approach from an operational deployment perspective, it appears that once the BI architecture is deployed on a single platform the BI team forgets the fundamental aspects of that architecture.

Physically storing all the data on a single platform appliance can be done, but often when companies deploy this architecture, they forget why they split the data into a hub-and-spoke. What often happens is the EDW and data mart-oriented data get mixed together or, worse, the data mart tables or

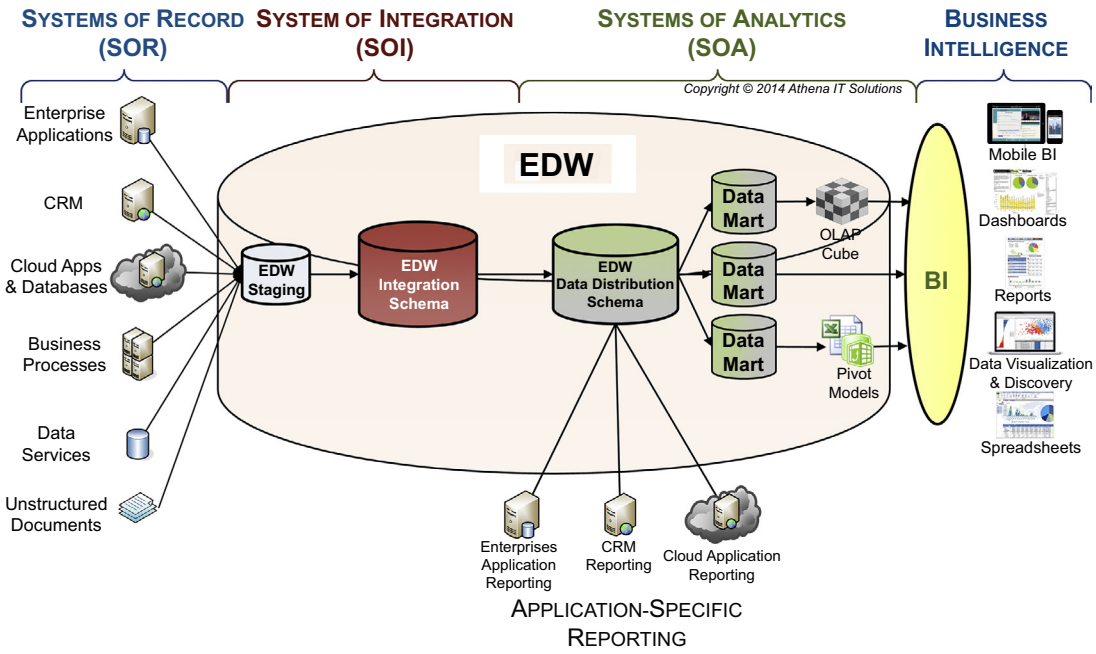

FIGURE 6.16

DI workflow—it's different this time.

cubes do not get built at all. Instead, people start trying to use the EDW schema for all reporting and analysis needs, eliminating distinction and functionality of the SOI and SOA. This shifts the architecture back to the old-school idea of a centralized DW, as depicted in Figure 6.17, trying to be all things for all purposes and generally getting none of it right.

Although the single platforms that enterprise are using are very powerful, that still does not negate the functional needs and benefits of separating the SOI and SOA data architectural components. I do not advocate physically separating them on different platforms, but to maintain their logical separation, and segment them on that platform by their schema and functionality. Although there are advantages to using a single platform, the risks are greater: the danger of misusing this technology and sending an enterprise BI effort back to the days of the EDW-only architecture, when it was harder for the business to leverage it for BI.

OPERATIONAL DATA STORE

No BI term has been muddied and misused more than ODS. There are consensus definitions for DW and data mart; however, an ODS seems to mean anything that isn't a DW or data mart. Whereas people understand how to architect, design, and build a DW and data mart, they are stumped when it comes to an ODS. IT departments, enterprise application vendors, and BI vendors have been very busy building ODS in the absence of a clear definition or agree-upon architecture.

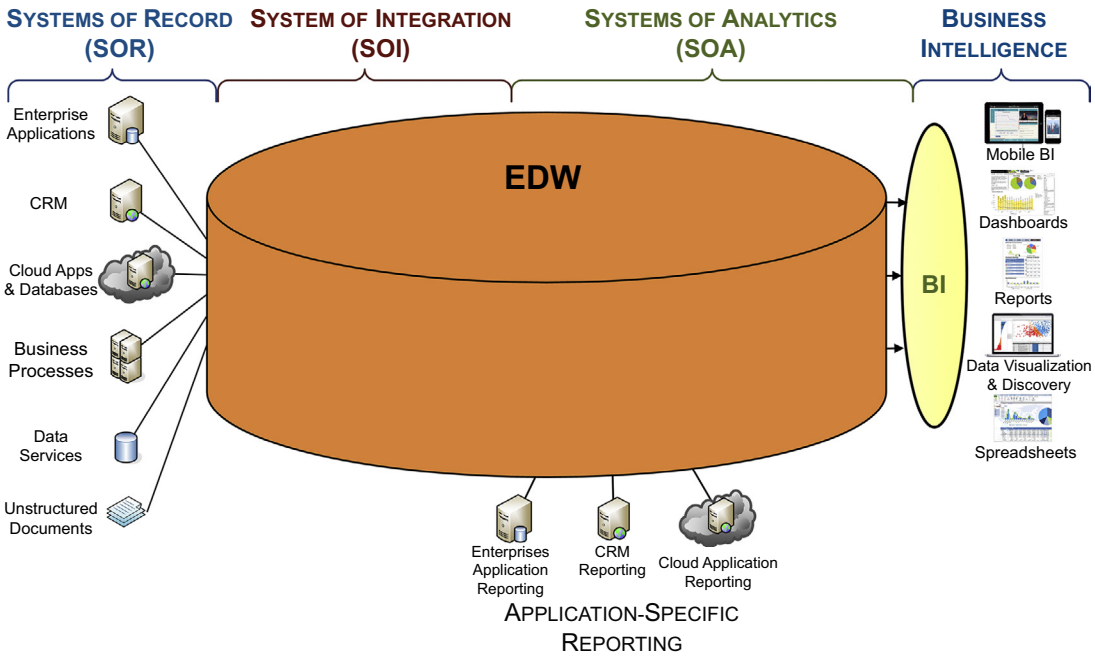

FIGURE 6.17

DI workflow—back to the future.

Although ODS designs vary widely, an ODS typically has these characteristics:

- Integrates data from multiple source systems
- Enables consolidated reporting
- Offers pre-built reporting or integration solution for enterprise applications
- Uses a data schema similar to source systems

ODS systems are usually distinct and separate from an enterprise's BI architecture. Because an ODS is created either by the IT group managing operation applications or the vendors that sell these same applications, they are usually extensions of operational applications that are not designed along BI architectural principles. Historically, the tools used for development were those used with the ERP systems, and the data schema was similar to the operational application. With different tools, database design principles, and groups developing the ODS and BI, it is no wonder they were not engineered to work in the same architecture and became data silos.

The methods used for building the ODS have changed dramatically over the years due to advances in technologies and industry knowledge in BI and data management. Many of the conditions that applied years ago when many ODSs were built don't apply anymore. The following sections will review the reasons for building ODS, examine whether those reasons are still applicable, and, if so, how these requirements should be addressed.

THE RATIONALE FOR AN ODS

ODS definitions have fallen short because they mainly explain what has been built rather than why it was built. As a result, people tend to define an ODS by its characteristics rather than the business or technical reasons for building it.

The reasons for building an ODS include:

- Providing integrated reporting across application.
- Establishing an SOR for dimensions.
- Creating a data hub for a business process.
- Achieving more frequent updates than an EDW.

The following sections discuss the reasons for building an ODS and how they fit in an enterprise's data architecture.

Integrated Reporting

ODS systems got their start by supporting reporting requirements across multiple transactional applications. They helped fill a gap in fulfilling DI across these multiple applications because it was not within the scope of the application vendors or the DW groups.

The vendors were too busy expanding or acquiring new functionality and modules, not focusing on integration and reporting across them. DW groups tended to draw a line between management reporting (what they did) and operational reporting (what they did not do). DW teams were occupied by building and maintaining their existing projects without taking on the burden of operational reporting. In addition, updating the data more often than once a day, or in real time, was more than many of them could handle. Thus, the group maintaining the applications managed integrated reporting activities. As a result, the worlds of ODS and DW diverged into different, overlapping, and counterproductive approaches.

Today, enterprise application and BI vendors use ODSs to develop integrated reporting solutions:

Enterprise application vendors have built ODSs into their products to meet their customers' ever-growing demands for reporting. A pre-built ODS is a way for these vendors to integrate across their own modules and their competitors' products. With so many of these applications created through mergers and acquisitions, an ODS is a very expeditious systems-integration approach for the vendor. These ODS systems are architected as the applications reporting solution and are typically designed with a schema similar to the operational applications. The solutions work well for application users and can be very cost-effective, helping avoid the need to build custom operational reporting systems.

BI vendors have developed corporate performance management (CPM) solutions that support both operational and management processes. The underlying ODS integrates data and enables real-time updates. These CPM solutions provide an alternative to custom-built solutions on top of enterprise applications.

SOR for Dimensions

When used as an SOR for categories of data, an ODS collects, integrates, and distributes current information and provides an enterprise view of it. Common examples are customer and product data. Business groups, such as divisions or subsidiaries, may be responsible for managing a subset of this data, but no single group is responsible for managing the overall customer or product lists.

Data categories such as customers or products are often referred to by different names (dimensions, reference data, or master data) based on your application background, as shown in Table 6.4.

Table 6.4 Data Category Descriptions

Role	Description
BI team & data modelers	Dimensions
Business people & application developers	Reference data (also cross-reference or look-up data)
ERP vendors	Master data

Regardless of what it is called, this data is typically scattered across multiple applications or source systems. For accurate reporting and analysis, an enterprise needs to make sure this data is consistent. For those in the BI, making this data consistent is referred to as conforming dimensions.

Historically, the BI group received de facto responsibility for making this data consistent. This is a great use of an ODS—where the ODS becomes the SOR for this consistent data or conformed dimensions. ERP vendors have recognized this need and offer master data management (MDM) solutions across their application modules through their own pre-built ODS solutions. When an enterprise's needs are greater than one ERP vendor's applications, a custom ODS may be required.

Establish a Data Hub for Updates

One of the most prominent purposes for an ODS is to provide DI, data distribution, and reporting to support specific business processes and associated data. The two most common examples are customer relationship management and budgeting/forecasting. Both of these involve integrating front-office applications rather than the back-office applications that are the mainstay of DW. With the front-office applications, many people need up-to-the-minute updates on information that is relevant to their

business function. With customer-facing applications, such as a call center, the business people need data on all aspects of their customers, but that data is spread across many applications.

The ODS provides the ability to act as a data hub to collect, update, and redistribute the data to people and the applications that they use. As a data hub, the ODS may be connected to various enterprise applications, DWs, and data marts, using them as both sources and targets (for data updates) in various business processes. This is similar to how budgeting and forecasting can be serviced by an ODS. The budgeting applications will use the ODS as the initial source of information, update the data through these applications, and then use the ODS to distribute these updates to the DW. In all cases, the ODS is used for the immediate reporting needs of its constituents.

Update Data More Frequently than a DW

A very common rationale used for creating an ODS is that the DW cannot handle either the data volumes or update frequencies required to support the operational reporting demands. This can be the case when a DW was originally designed to update data daily, but doesn't have the hardware capacity to expand, or if it does, the performance would degrade. On the other hand, even if the capacity exists, handling operational data may be outside the charter of the DW group, and should be assigned to the application group.

ODS REEXAMINED

Several advances in technologies and BI knowledge have significantly affected the "why" and "how" of building an ODS. The following sections will look at the previous reasons for using an ODS to determine if they are still applicable.

Integrated Reporting

Although there are still many ODS-based integrated reporting solutions offered by enterprise application and BI vendors, many of the early adopters have replaced their operational data stores with BI architectures such as DWs, data marts, and cubes using the same technologies recommended in Chapter 7. They have done this for several reasons:

- The evolution of BI best practices and the technologies to support them
- Industry pushback against the technology and data silos that the ODS-based solutions created
- A cross pollination of BI knowledge and a desire to offer an integrated BI architecture as mergers and acquisitions have blended ERP, BI, and DI vendors together.

There is still a tendency for IT vendors or IT groups who work exclusively with operational systems to initially design ODS-based solutions for integrated reporting, but those typically become stop-gap approaches until they can shift to a BI architecture.

SOR for Dimensions

There are two approaches available based on the business and technical requirements:

First, if the data volumes, frequency of updates, and type of data cleansing needed justify it, then an enterprise should invest in an MDM solution that manages data on their customers, prospects, products, or employees. These types of solutions and their architectures are discussed in Chapter 5.

Second, if a full-fledged MDM is not needed, the enterprise should build a solution based on the principles of conforming dimensions using a hybrid dimensional-normalized schema. This solution can be built in either the overall BI architecture within the EDW integration schema or in a schema within an enterprise application. The recommended approach is within the EDW, but there may be conditions, such as the hosting environment, that push an enterprise to host that solution in an enterprise application.

Establish a Data Hub for Updates

DI used to be limited to batch-driven ETL processes. Because of this, people used ODS data hubs to enable near real-time data updates. Today, however, significant advances in integration technologies have replaced this approach:

- DI has expanded to include many real-time processes such as data services that can query and update data.
- Data virtualization has become much more robust: it handles larger volumes, a variety of data sources both structured and unstructured, and it can be incorporated into a BI architecture.

Update Data More Frequently than a DW

In the past, creating an ODS to handle real-time or near real-time updates was a pragmatic approach because ETL and database technologies were not robust enough to update a DW with that frequency. However, these technology reasons for creating an ODS do not apply because advances in infrastructure, DI, and databases can support real-time DW updates if the enterprise has the business requirements and is willing to make the required investments.

There are conditions when real-time or near real-time updates need to be segmented from the core EDW integration schema. One of these conditions occurs when the dimensional and transactional data has inconsistent times, causing referential integrity issues. These inconsistencies are addressed when the underlying applications are synchronized, but until that happens they can remain inconsistent throughout the day. Based on business needs, the BI environment supporting the more traditional periodic updates should continue to provide analytics to the business while the business processes that need to use the real-time updates should be aware of and handle the inconsistencies.

Segmenting real-time data from the EDW integration schema is a prime scenario where an ODS-like application is justified. The ODS should adopt a hybrid dimensional-normalized model and be adopted into the BI architecture.

A caveat is that before building this solution, an enterprise should consider if data virtualization would be a better alternative.

Sourcing Unstructured or Semi-structured Data for Analytics

An ODS has often been used as a way to perform a non-traditional EDW function or capability. Supporting Big Data analysis of semi-structured or unstructured data is the latest capability that does not lend itself to traditional EDW schemas today. For example:

- A NoSQL database may be used to capture semi-structured or unstructured data
- Textual analysis is performed on the captured data
- Results of textual analysis are stored in an ODS that might use any of a variety of database technologies based on its intended usage

ODS IS DEAD, LONG LIVE ODS

Much of the technological rationale for building ODSs has dissolved, and advances in BI best practices provide a blueprint on how to address the business requirements that an ODS met.

That is, of course, for the ODSs that were built in the past. An ODS was always a pragmatic approach to addressing the shortcomings in the technology and state of BI expertise at that moment in time. There will continue to be new data analysis requirements that cannot be addressed at the moment, and it is likely that an ODS will arise to meet that need. Already, ODS-like structures are filling some of the gaps in Big Data and cloud-based data solutions.

When asked what an ODS is the answer will likely be "We will know it when we see it." And just as likely, the industry will respond to fill the gaps that the ODS pragmatically plugs.

REFERENCES

[1] Kimball R, Ross M. The data warehouse toolkit: the definitive guide to dimensional modeling. Wiley; 2013.
[2] Inmon WH. Building the data warehouse. Wiley; 1991.