The data consolidation pattern specification helps data and application architects make informed architectural decisions and improve decision guidelines. In this article you will see how you can apply the pattern in the SOA context. The primary business driver for the data consolidation pattern, also referred to as the data population pattern, is to gather and reconcile data from multiple data sources before this information is needed. To do so, it extracts data from one or more sources, transforms that data into the desired target format, and loads it into some persistent data target. The prepopulation of a persistent target is a key differentiator between this pattern and the data federation pattern covered in Part 1 of this series.

View more content in this series

Introduction

Business growth forces companies' IT capabilities to evolve with changing business demands. New applications are introduced to support innovative requirements. Existing information is processed and analyzed in new ways to gain even more insight into critical business challenges. Companies merge and acquire other companies, further accelerating their growth into new areas. Unfortunately, the information/data landscape does not always evolve in a strictly controlled and organized way to support this growth. Islands of redundant and inconsistent information arise. The same data is represented in many different ways by different applications in order for each one to achieve maximum efficiency in a specific area.

Companies are adopting a Service-Oriented Architecture (SOA) to address a wide range of challenges, such as the need to reduce the cost of system integration and optimize the reuse of existing information and functionality. One of the critical steps toward adoption and implementation of SOA is the identification of the most critical business functions (services) and their design. It is common practice to focus on services that can be leveraged by many consumers across and beyond the enterprise. The services in this scope will most likely need to draw data from a wide range of diverse systems that each hold different pieces of the required information. For example, most companies do not store customer information in a single place. This becomes a significant
problem if it is not clear where to get the information and which system holds the most current and accurate information. Without that knowledge, it is impossible to implement a service that returns a consistent set of customer-related information.

This article describes the data consolidation pattern as one means to integrate information from diverse sources. The relevant information is first gathered from various sources. The data is then processed to resolve conflicts and to create a common structure representing the target model. Finally, this transformed information is then applied to a target data store.

**Value proposition of the data consolidation pattern**

**Transparency of underlying heterogeneity**

The consumer sees a single uniform interface. The consuming application of the pattern does not need to be aware of:

- Where the original source data is stored (location transparency)
- What language or programming interface is supported by the source databases, for example, whether XQuery or SQL is used, or what dialect of SQL the source supports (invocation transparency)
- How the data is physically stored (physical data independence, fragmentation, and replication transparency)
- What networking protocols are used (network transparency)

**Performance and scalability**

The data consolidation pattern decouples the data-integration task from the data-access task. Accessing the target database does not require the execution of a data-consolidation process. Typically, the consolidation process is scheduled to occur daily, weekly, and so on -- independently of the access by a consumer of the target data. Because the required data is already gathered in one location, the highest level of performance and scalability can be ensured for data consumers.

**Single version of the truth**

This approach applies powerful capabilities to resolve conflicts when data is integrated from heterogeneous sources. Services can then draw from this consolidated repository and satisfy high data-quality requirements.

**Reusability**

After applying the data consolidation pattern to a particular integration scenario, the result of the consolidation process can be provided as a service to multiple service consumers. For example, a scenario might require integrating financial information from multiple regions. When the data consolidation pattern is applied, the disparate data is consolidated into a single place, which is then exposed through a financial dashboard. The same consolidated data can then be leveraged through information services to other consumers, such as automated processes for standard claims applications or client-facing Web applications.
Improved governance

Governance is a key underpinning to the SOA life cycle. Patterns enhance the governance process by reinforcing common practices with predictable outcomes. Reuse of proven flexible patterns in the development and creation of systems can both ensure consistency and quality and reduce maintenance costs by having a single source to update with changes.

Context

This pattern has been deployed in a variety of scenarios in a traditional and non-SOA context over an extended period of time. Based on the increasing interest of SOA, we see new opportunities to apply this pattern in the SOA context.

Traditional, non-SOA context

The most typical scenarios in which the data consolidation pattern has traditionally been applied are:

- **Application migration**: Application migration takes place when an existing legacy system -- for example, a homegrown customer relationship management (CRM) system -- needs to be replaced by a new application for business or technical reasons. Data consolidation supports the application migration process to move the data from the legacy environment into the future application's database and to apply any required restructuring of the model and the data itself.

- **Application consolidation**: One of the tasks in consolidating applications -- for example, reducing the number of a variety of enterprise resource planning (ERP) systems into a single or very limited number -- is to consolidate the underlying databases. That means that the data from the variety of existing legacy systems has to be merged into the consolidated database(s).

- **Decision support**: Many decision-support scenarios such as those that address financial analysis and reporting require access to data distributed across a wide range of sources. The quality of the decisions depends on the quality and the comprehensiveness of the underlying information. Therefore, distributed data needs to be integrated and made available for extensive analysis. In many cases, historical (copy) snapshots are taken to review trends over a period of time. Data consolidation helps to provide companies with this single version of the truth from a wide range of sources. A data warehouse to support decision making is a typical example of the use of a data consolidation pattern.

- **Master data management**: Master data management aims to decouple master information, which is defined as the facts that describe the core business entities, such as customer and product, from individual applications. The creation of this master data, or single version of the truth, is accomplished through a set of disciplines, technologies, and solutions used to create and maintain consistent, complete, contextual, and accurate business data for all stakeholders of the information. The driver behind a master data management initiative is a situation where the master data resides in many isolated systems, stored and maintained in different formats, resulting in a high degree of inconsistency and completeness. In order to produce an accurate and consistent set of information, which can be managed in a central
master data management system, data needs to be gathered, transformed into the master data model, and consolidated into the master data repository.

All of these scenarios share a common thread:

- The source information is distributed across multiple heterogeneous and autonomous systems
- The source information can exist in inconsistent or incomplete formats
- Inconsistent rules are applied to the source data
- The flexibility to change information sources and formats is fairly limited

The source data must be consolidated into a single persistent target to address those challenges by integrating the data into a common and consistent format. The core functionality of the data consolidation pattern addresses this requirement through the three component activities of gather (extract the data from the sources), process (transform the source data to match the model that defines the target), and apply (load the consolidated and harmonized data into the target data store or system). This is illustrated in Figure 1.

Figure 1. Traditional data consolidation pattern

SOA context

The SOA context presents many similar challenges to the traditional context, so we believe it’s important to reuse these proven existing approaches and enhance them to apply them within SOA.

First SOA use case:

The first SOA use case is an extension of the scenarios described in the traditional context above. In this use case, the consolidation process for the target data is now exposed as a service. For example, a master data management solution might be centered on part information or vehicle information at an automotive manufacturer. Because of the importance of part/vehicle information, many consumers will need to access this data from a consolidated master data management system. A service such as getVehicleData would be an instantiation of the consolidation pattern.
that implements a reusable service. This service can then be accessed across and beyond the enterprise, for example by employees as well as external parts distributors.

Exposing this information through services increases the potential reuse of this implementation and therefore can reduce the overhead and inconsistencies associated with the derivation, transformation, and variation in source formats that are frequently introduced when multiple consumers (implementers of the target systems) need to perform this integration task separately and redundantly. In the SOA approach the enterprise service bus (ESB) brokers the messages (service request and response) between a variety of consumers and the information service provider, as illustrated in Figure 2, thus enabling the consistent and standard invocation of the service.

**Figure 2. SOA access to consolidated data**

Enabling information integration services within a SOA requires additional functionality that encapsulates information access within a service-oriented interface. This is accomplished through information service enablement. The purpose of this component is to expose consolidated data in a service-oriented interface. For example, the consolidated vehicle data might be stored in a relational database. Through the information service enablement component, this relational vehicle data can be exposed as a service -- for example, defined by Service Component Architecture (SCA) or Web Services Definition Language (WSDL). The service that implements access to vehicle data can then be shared across and beyond the enterprise.

**Second SOA use case:**

The second SOA use case illustrates a situation in which a consumer invokes the consolidation process. Traditionally, the consolidation process runs on a relatively fixed time schedule, most often during maintenance windows on a weekly or daily basis. The consolidation is decoupled from business processes that often run in a less rigid time schedule. In the SOA context, a certain step in a business process or an application can directly invoke the consolidation process. Two examples of this use case are:
• "Refresh my DataMart Now" might be a button available to a business analyst who only wants up-to-date information "when needed." Automatic refreshes are problematic because of the point-in-time analysis being performed on financial data. The updates should only be as "real time" as dictated by the subject-matter expert -- not necessarily when technically feasible. Invoking the data consolidation process through such a button provides a solution to this requirement.

• A major pharmaceutical company uses the data consolidation pattern in a SOA context to support the collection and review of remote lab-testing statistics. Standardizing the collection of lab data helps to shorten the already long and expensive life cycle of pharmaceutical offerings. In this application of the pattern, individual labs invoke a consolidation service once they have inserted research detail into an isolated transactional system and placed graphics (presentations and JPG files) into a central directory for approval. The consolidation service collects the statistics from a given lab and stores them in the company's centralized tracking system. Statistics are generated by multiple disparate tools, which in turn have unique data stores. Prior to implementation of the pattern, statistics and supporting materials were collected less effectively by various manual, homegrown electronic and paper-based systems.

Figure 3 illustrates how an activity in a business process (called "invoke" in the figure) sends a request -- optionally through an ESB -- to the component that realizes the information service enablement. This component takes the service request and invokes the consolidation process. Subsequently, the data is gathered from the sources and processed, and the result is applied to the target.

**Figure 3. SOA accessible and reusable data consolidation processes**

Third SOA use case:

The third SOA use case represents a combination of two patterns: data consolidation and data event publishing or change data capture, as in Figure 4.
Many companies are challenged to achieve effective inventory management. Part of the problem is that inventory-related information resides in many heterogeneous databases. However, in order to optimize inventory, the information needs to be accessible in an integrated and consistent manner. For example, different part numbers for the same part need to be resolved when information is consolidated for inventory access and analysis. Some of the inventory-related information is changing very frequently -- for example, for products in high demand -- while other data remains unchanged. This situation requires consolidation of distributed, heterogeneous, and frequently changing information into a single repository.

Data event publishing and "trickle feeding" of data to a target repository is another important context for applying the data consolidation pattern. Some of the key drivers for using this approach are:

- Target systems are synchronized with the sources in a near-real-time manner. This can be critical for decision-support applications or cooperating operational systems. Consumers of the target see immediate results.
- This pattern precludes the need for long sweeps of source systems during batch windows. Such sweeps can degrade performance of the source, affecting other applications.
- Only "changed" information is sent across the network and passed through extensive transformation processes, thus lessening the burden on the network and systems performing manipulation.

Problem statement

SOA consumers request a service that requires access to information from multiple heterogeneous sources. The sources have been designed, developed, and evolved independently so that they
have significantly different representations of the same types of data. The heterogeneity can occur on an instance level -- for example, lack of a common key (because of different formats), or on a model level -- for example, the same real-world entity is modeled in a different number of database entities. The SOA consumers must not be aware of this underlying heterogeneity but must be able to access the integrated information transparently.

Many situations in which this pattern is applied require a high level of availability of integrated information. Often, the source systems are constrained because of resource utilization and the limited flexibility for application changes. At the same time, rather complex data-processing operations need to be performed in order to provide the requested service.

**Solution goals**

The goals are to:

- Integrate information from sources that have possibly a high level of heterogeneity and support read-only access to this integrated information with a high level of data availability, scalability, and performance.
- Provide extensive transformation capabilities in order to resolve conflicts between sources and in order to restructure source data into a desired target data mode.
- Decouple access to the integrated target data from the process of integrating and transforming data from sources into the target in order to allow for scalability and performance.

Enable scenarios that require updates to consolidated data, such as operational master data management systems, to combine this pattern with other approaches that propagate changes in the target system back to the sources and thus keep those systems synchronized.

**Solution description**

The data consolidation approach has three major phases. In the first phase the consolidation server -- the component that implements the data consolidation pattern -- gathers (or "extracts") the data from the sources. Next, the source data is integrated and transformed to conform to the target model, possibly in multiple operations. Last, the consolidation server applies the transformed data to the target data store.

This process can run (repeatedly) on a time schedule or it can be invoked (repeatedly) as a service from a business process or any other service consumer. After the integrated data is loaded or refreshed in the target, the consolidated information can be exposed as a service to consumers.

**Design-time characteristics**

The key task during design time is to specify the data flow from the sources to the target -- that is, how to restructure and merge source models into the target model. It is assumed that the administrator or developer of the data flow, when applying the data consolidation pattern, has a detailed understanding of available access interfaces to the sources, the semantics and correctness of the source data model, and its integrity constraints. It is further assumed that the
target data model is defined. If these assumptions are incorrect, the consolidation pattern must be combined with other approaches, such as data profiling and data modeling, that resolve these open issues.

Based on those assumptions, the developer defines the set of operations that can transform the source data into the target data corresponding to the source and target models. Implementations of this pattern vary in the range of transformation operations supported and how extensible those operations are. Implementations tend to use a set of the most typical operations, such as lookups, joins, and filters. Additional implementations provide powerful mechanisms to extend this set of operations for customer- or project-specific needs. Using these operation capabilities during the design phase, the data flow (that is, the sequence of those data processing and transformation operations) is defined. This data-flow specification is then deployed to the consolidation server and controls what data is extracted, how it is transformed, and how it is applied to the target.

The data-flow specification is a specific type of metadata. Combining this metadata with other related metadata can support other applications (ones that are outside of the scope of this pattern) such as impact analysis and business glossaries.

Implementations of the consolidation pattern -- more specifically of the information service enablement component (see Figure 2) -- vary in the level of tooling support and configuration options that assist the administrator or developer in generating a service interface for the invocation of the consolidation process.

Functionality of the information service enablement component can also help to map a service interface to a query that accesses the data in the consolidated database.

**Run time**

The consolidation server implements the data-flow specification that is defined during design time. The execution of the data-consolidation process is started based on a defined time schedule or through a service invocation. The first step in this consolidation process is to access the source systems in order to gather the relevant information. Typically, consolidation servers use a set of source-specific connectors that may also be referred to as wrappers. Each connector is designed for a specific source type -- such as DB2 or Oracle -- to gather the information and deal with source-specific interface characteristics most effectively. For that reason, the connectors support different interfaces toward the sources and provide one common interface to the core consolidation server.

After the data is gathered through the connectors, the core of the consolidation server processes the data according to the data-flow specification. The consolidation server resolves conflicts among the source data streams, joins data together or splits it apart, transforms the data to correspond with the target model, and processes the data possibly by further lookups to other sources. As part of this process, the data that is gathered from the sources and being transformed might need to be persisted temporarily in so-called staging areas.

Once the structure of the processed data conforms to the target model, the consolidation server applies the data to the target, possibly using target-specific connectors again.
Although a consolidation server can process single records, most implementations are targeted to move large amounts of data from various sources to one or more targets. This is often referred as *bulk data movement*. Some products that realize this pattern exploit parallelism to process the data more effectively.

**Considerations**

When applying the data consolidation pattern it is important to understand how it impacts the following nonfunctional requirements.

**Data security**

The security configuration in the target database -- on top of which the services are defined -- is independent of the sources. As we stated previously, this pattern is most frequently applied to moving data in a batch/bulk mode from sources to the target. This process is often applied to the complete data set in the sources -- that is, without security restrictions. Often, the target is created when the consolidation pattern is first applied so that no access controls preexist and possibly need to be defined. Each data source can have its own security restrictions, which might need to be addressed in order to allow the data to be accessed and retrieved appropriately.

Because of the heterogeneous and distributed nature of this environment, some challenges regarding single sign-on and global access control might arise that are outside the data consolidation pattern's scope. In order to address those challenges, architects will need to combine the data consolidation pattern with other security-related patterns.

**Data latency**

Generally, the data latency or data currency is dependent on the refresh cycle of the data consolidation process.

Historically, the data consolidation process is triggered by a time schedule on an infrequent basis, such as weekly or daily. After the consolidated data is applied to the target, it is traditionally not refreshed before the next cycle. More recently, this latency issue has been improved by aligning the consolidation phase with the appropriate business process. As shown in Figure 3, an activity in a business process or an application can invoke the consolidation process. This allows refreshing the consolidated data through a service just before the data needs to be consumed.

Combining the data consolidation pattern with the data event publishing pattern, as shown in Figure 4, further improves the data currency. Changes in the sources are captured as they occur and are then immediately consolidated into the target.

**Source data volatility**

The more frequently data in the sources changes between the refresh cycles of the consolidation process, the more stale the data becomes in the target. In order to increase consistency between the source and target data, source changes can trigger the consolidation phase to be executed through data event publishing. Alternatively, the consolidation process can be invoked through an application or an activity in a business process that is aware of source changes. However, a more
frequent refresh cycle can have a negative impact on resource utilization. In particular if it is not coordinated with the demands of the consuming application, data might be frequently refreshed in the target without being consumed, thus using the resource less efficiently.

Data consistency and quality
The consolidation approach is especially advantageous for providing powerful mechanisms that can address situations in which source data has a low level of data quality and consistency. Complex data cleansing, standardization, and transformation operations affect only the duration of the consolidation process but do not affect the response time or scalability of the service request to the target.

Data availability
The availability of integrated data in the target depends solely on the availability of the target system, such as a database. The process of consolidating data and populating it in the target is decoupled from the request flow when a consumer accesses the data in the target. From a consumer perspective, accessing consolidated data in the target system has the same availability characteristics as accessing any other data in this system. Therefore, any approaches to increase data availability can be applied in combination with the data consolidation pattern. Since the data consolidation has only a single target, it is relatively easy to apply technology to improve availability -- clustering, for example. This pattern is a preferred approach if high availability of data is required.

Impact of model changes on integrated model
When any of the source models change, data-flow specification and possibly the target model will need to be adjusted. If the target model needs to be modified, the target data will need to be adjusted accordingly. Depending on the required changes, this can have a minimal or significant impact on the availability of the service.

Frequency of transaction execution
The frequency of service requests against the consolidated target is only determined by the ability of the target database and the information service enablement component to handle those requests. Since the target is created specifically to support those service requests, this pattern is a preferred approach for requirements of highly frequent transaction execution.

The ability of the consolidation server itself to execute a data movement transaction at a high rate is determined by the rate at which the consolidation server can access the source systems and the source systems can respond to provide the data. Because of the decoupled approach we discussed above, this does not have an impact on the frequency with which the service request against the target can be executed.

Transaction concurrency
Efficient management of concurrent access (of service requests to the target) is determined by the performance characteristic of the target database server. This is due to the decoupled approach of this pattern.
Performance/transaction response time

The transaction response time of a service request against the consolidated target is primarily determined by the characteristics of the target database server. This due to the decoupled approach of this pattern.

Create-Read-Update-Delete profile

The data consolidation pattern moves data unidirectionally from the sources to the target. External changes to the target data store are outside the scope of this pattern. As such they are not propagated back to the sources by this pattern and can be overwritten during the next refresh cycle of the target. Therefore, this pattern is typically applied only in situations where read-only access to the target is sufficient.

Data volume per transaction

The data that is exposed in the service request to the consolidated target store is retrieved directly from the target database. Therefore, the performance characteristics of this approach are determined only by the target database server.

The consolidation process of moving data from the sources to the target is designed to support large amounts of data. Because of the decoupled nature of this approach, service requests to the target can be handled efficiently even for large data volumes. The same is true for the data movement process itself.

Solution delivery time

Product implementations of the data consolidation pattern frequently provide highly sophisticated tooling support to specify the mappings (data flows) between the sources and the target. Many of these implementations have predefined (data-flow) operations that are provided out-of-the-box with the products. This allows the implementer to apply this approach efficiently in a short period of time.

However, this pattern is often applied when data sources with significant differences in the structure of data need to be integrated. This can require iterative refinements of the data-flow specification and applying the specification in test environments to prove the correctness. Companies can experience relatively long development cycles when applying this approach -- not because of the characteristics of this approach but because of the characteristics of the problem.

Skill set and experience

Most existing implementations of the consolidation pattern have a tooling approach, which requires product-specific knowledge when defining the mappings. Developers need to understand these product-specific approaches. They also need to have knowledge of database concepts or DBA experience in order to understand the implications for the source and target database when designing this solution. When exposing integrated information as services, developers also need to understand SOA concepts, standards, and technologies.
Reusability

Logic and metadata used to define data access and aggregation can be reused across different projects.

Cost of maintaining multiple data sources

Following data consolidation it is possible either to leave the original data sources intact, or to retire the sources once the data is moved to the target in the case of a migration. As described in the use cases (see Context), this new target system often meets additional business requirements such as providing the single version of the truth and additional insight. When the pattern is used to migrate from (that is, replace) existing legacy systems, moving -- and possibly consolidating -- the data is just one step in the overall migration process. The overall process also needs to address, for example, the migration of business and application logic. Although the data consolidation pattern cannot solely address application migration, it is an important component in that it can move the data from the legacy system to the future platform. Once the overall migration process -- including data, logic, and processes -- is completed, the cost of maintaining multiple data sources can be reduced by eliminating the legacy system(s).

If one of the goals of the project that implements or uses this pattern is to create a new data repository, an incremental cost might associated with the management of the new data store. However, that is not a side effect of the pattern implementation but rather a result of the larger project that may use this pattern.

Cost of development

The development costs depend largely on the complexity of the integration task. The costs can be low if the data sources have similar data models and only simple transformation operations are required. The more complex the mapping between sources and target becomes, the higher the implementation development costs, which are associated with the iterative development and testing cycles necessary to address the complexity.

Type of target models

The data consolidation pattern does not require a specific target data model. In this article, we have focused on the data consolidation pattern for structured data. Most of the structured data is maintained today in relational systems. Therefore, most deployments of this pattern move data to a relational target database.

Assured delivery / logical unit of work

Although the consolidation approach does not inhibit assured delivery, most current implementations of the data consolidation pattern do not guarantee assured delivery of data movement between the sources and the target. If for some reason the consolidation process is interrupted, for example because of server failure, some of the data might have been already moved, some of the data might be in the process of being moved, and some data might not have
been moved. The system should either have the capability to restart at the point of failure or have compensation logic that enables undoing incomplete updates. As in any failure situation, SOA or not, this does not preclude the necessity that in some cases the architects, administrators, and developers will have to analyze the root cause and determine the recovery process.

**Resource utilization**

The consolidation server utilizes resources -- that is, processing power on the consolidation server, the source servers, and network capacity -- when it moves data from the sources to the target. The level of utilization is determined by the complexity of the transformations, the number of sources to be accessed, and the volume of data to be processed.

**Transformation capabilities**

The implementation of the consolidation pattern should address the need to resolve almost any differences found in structures between the source and target data. An important consequence of highly complex transformations is an elongated data-movement process that is due to the complex transformation processing.

**Type of source model, interfaces, protocols**

Data consolidation addresses the problem of integrating data from heterogeneous source models and includes techniques to map those different source models into the common model at the target. Product implementations of the data consolidation pattern vary in the range of source models they can integrate, but for the most part, the data consolidation pattern removes the complexity of source models, interfaces, and protocols so developers only need to care about one model, interface, and protocol.

**Scope / size of source models**

The size of source models, the number and type of attributes, and the complexity of defining the transformation can be time-consuming for the data analyst, architect, and implementer. These factors can impact the time needed to implement the pattern as well as the time and resources to perform the consolidation. Standard project scoping and definition practices should address the degree of complexity associated with the data transformations when assessment of the effort, duration, and cost associated with the project is made.

**Impact of consolidation server workload (transaction volume) to sources**

An impact analysis on the source systems should be performed to understand the impact of the requests on the service levels that these sources are already committed to providing. This should be a standard step in the development methodology and is not unique to this implementation. The movement process can be coordinated so that it has minimal impact to the sources, for example during maintenance windows to minimize the impact to operational source systems. This need has to be balanced against the delivery timeliness and latency requirements for the consolidated target.
Conclusion

This article has presented the data consolidation pattern as an approach to gathering data from multiple sources, to processing and transforming this data, and then to applying it to a single target. Service consumers in a SOA often need access to heterogeneous and sometimes conflicting information. Data consolidation can integrate the data and resolve conflicts, and therefore can create the single version of truth required. This consolidated information can then be exposed through a service.

Focus areas to apply the data consolidation pattern

• Integrating data from a wide range of sources with a high degree of heterogeneity: This approach has powerful capabilities to resolve the conflicts and merge the data together. The data consolidation pattern is often combined with the data cleansing pattern so that data-quality issues can be addressed during consolidation.

• Providing integrated information for consumers that demand high data availability, high level of concurrent access, high scalability, and performance: The data consolidation pattern materializes the integrated information in a new target copy that the consumers can access independently of the transformation and integration process.

Risk area to apply the data consolidation pattern

Real-time access to distributed data that is frequently changing: Addressing this scenario with data consolidation requires frequent movement and consolidation of the source data. If the consumer rarely needs access to this integrated information, this approach might not be as cost effective as other approaches and might not deliver the data as up-to-date as the application expects.

Product mapping

The following IBM products implement this pattern:

• IBM® WebSphere® DataStage Enterprise Edition (also part of the WebSphere Data Integration Suite portfolio offering and of the IBM Information Server portfolio offering) is a high-volume data-integration platform for data cleansing, transformation, and relocation. Complex data flows in WebSphere DataStage are developed using a graphical "dataflow"-oriented paradigm that promotes reuse and enhances developer productivity. Parallel processing capabilities, such as support for dynamic repartitioning, parallel databases, and grid configurations, enable WebSphere DataStage to manipulate massive quantities of data in short time frames. Sources and targets include relational database management systems, ERP systems, mainframe legacy systems, XML, and proprietary data formats. The extensible platform that WebSphere DataStage is built upon runs on UNIX, Windows, Linux, and zSeries environments, and it includes a comprehensive metadata layer for management and control of business rules for enhanced data governance and entity tracking.

• WebSphere Information Services Director (also part of the IBM Information Server portfolio offering) exposes information-management capabilities as services. It packages information-
integration logic, cleansing rules, and information access as services. This insulates the
developer from the underlying provider of this functionality. Most relevant to this article is its
capability to expose WebSphere DataStage jobs through a service-oriented interface such
as EJB, JMS, or Web services. This product provides the foundation infrastructure (including
load balancing and fault tolerance) for information services. It realizes the information service
enablement component illustrated in Figure 2, Figure 3, and Figure 4. The WebSphere
Information Services Director is built on the same powerful metadata infrastructure as
WebSphere DataStage.

Acknowledgments

We would like to thank Jonathan Adams, Lou Thomason, and Fan Lu for their support in writing
this article and in developing this pattern.
Related topics

- **Patterns: Information Aggregation and Data Integration with DB2 Information Integrator** is an IBM Redbook that documents and prototypes the role of information-integration technologies and architecture in IBM's Patterns for e-business using a typical customer insight e-business scenario.
- Consolidate and move your data with WebSphere DataStage (see Product mapping for further details).
- Enable consolidated data or data consolidation processes as services with WebSphere Information Services Director (see Product mapping for further details).
- The IBM SOA Web site offers an overview of SOA and how IBM can help you to get there.

© Copyright IBM Corporation 2006
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)