Third Nature Technology Report

The Role of Open Source in Data Integration

January 2009

Mark R. Madsen

TABLE OF CONTENTS

Introduction

Open Source and the Future of Data Integration

Data integration(DI) has seen minimal automation over the past decade despite many technology advances. Most companies still hand-code data integration between applications (operational data integration) using techniques that would be familiar to a programmer from the 1980s. In business intelligence 40% of the Extract Transform and Load, or ETL, processes are still hand-coded.

In the next few years it's likely that there will be stable to declining investment in new applications due to economic factors, but increased need for data integration technology. Integration consumes a significant portion of the IT budget and is coming under heavier scrutiny as a cost to control.

New vendors are addressing this gap in data integration, but face adoption challenges in IT. Data integration is a developer task and an infrastructure item, making new tools hard to justify. Fortunately, there are open source products in this market capable of supplying the much-needed automation.

Open source data integration tools can provide the cost advantages of hand-coding with the productivity advantages of traditional data integration software. They are established in the developer tools market which has been the traditional stronghold of open source software. Expect open source to be a key component of data integration (and especially of operational data integration) in the near future, similar to the way it is a key component of application development environments today.

Spending Priorities Emphasize Need for Data Integration

Business intelligence appears consistently as the top item in surveys of business and IT management priorities. More business intelligence means greater need for data integration tools. BI market surveys show that roughly 40% of companies are hand-coding their ETL processes, leaving room for growth.

A *CIO Insight* IT spending survey shows that standardizing and consolidating IT infrastructure is the number one priority in the coming year for large firms, and number two in medium and small firms. Across all firms, improving information quality shows up as the number three priority.

Results from a survey by Oracle on data integration showed that 30% of their customers are buying tools for operational data integration today. The top needs of these customers were low-latency data access, doing migrations, and creating data services.

These statistics clearly highlight the new focus on data integration. Only 60% of the business intelligence market is using data integration tools, so there is still room for growth. With 70% to 85% of companies still hand-coding operational data integration, it's clear that this is an area ready for automation, with early adopters already using these tools.

The Drive Toward Open Source Data Integration

Open source has become a standard part of the infrastructure in IT organizations. Most are using Linux, open source development tools, and many are running open source databases. The majority of enterprise web infrastructure is built using open source.

This growing familiarity with open source led to increased adoption rates across all categories of tools and applications. Venture capital flooded into open source startups over the past several years resulting in an explosion of enterprise-ready tools and applications.

Open source data integration vendors are creating challenges for both traditional vendors in the DI market who are trying to introduce new tools, and for new non-open source vendors of operational data integration tools. The existence of open source tools in a market raises barriers to entry that are hard for vendors to address. This is the scenario that played out in the web server, application server, and Java development tools markets.

Enterprise customers are demanding project-sized data integration tools that can be scaled up to enterprise use. They don't want complex, expensive DI products that are not a fit with the distributed nature of the application environment. With such a large market need, the future direction of data integration is sure to have a large open source component.

Understanding Data Integration

The Difference Between Application Integration and Data Integration

Data integration (DI) and enterprise application integration (EAI) are not the same thing, though vendors sometimes obscure the difference to broaden the appeal of their tools. Application integration focuses on managing the flow of events (transactions or messages) between applications. Data integration focuses on managing the flow of data and providing standardized ways to access the information.

Application integration addresses transaction programming problems, allowing one to directly link one application to another at a functional level. The functions are exposed to external applications via the tool's API, thus hiding all applications behind a common interface.

Data integration addresses a different set of problems. DI standardizes the data rather than the transaction or service call, providing a better abstraction for dealing with information that is common across systems. DI tools abstract the connectors, transport and more importantly manipulation – not just the system endpoints. When done properly, DI ensures the quality of data as it is being integrated across applications.

The type and level of abstraction are what differentiates the two classes of integration. EAI tools are a transport technology that requires the developer to write code at the endpoints to access and transform data. These tools treat data as a byproduct. This makes functions reusable at the expense of common data representations.

Data integration tools use a higher level of abstraction, hiding the physical data representation and manipulation as well as the access and transport. The tools provide data portability and reusability by focusing on data and ignoring transaction semantics. Because they are working at the data layer there is no need to write code at individual endpoints, and all data transformation and validation is done within the tool.

The key point in differentiating DI and EAI is to know that there are two distinct types of integration with separate approaches, methods and tools. Each has its role, one for managing transactions and one for managing the data transactions operate on.

Operational Data Integration vs. Analytic Data Integration

There are two different ways of using data integration tools based on the type of systems being integrated: transactional applications or business intelligence systems. These uses affect the approach, methods and tools that are best for the job. "Extract, transform and load" or ETL is the term used in analytic systems. The industry is settling on the term "operational data integration" or OpDI when referring to data integration for applications.

Business intelligence has been the primary driver of data integration products for the past decade. BI systems are most often loaded in batch cycles according to a fixed schedule, bringing data from many systems to one central repository. They have relatively large volumes of data to process in a short time, but have little concurrent loading activity. Most products were originally designed to meet the specific needs of the analytic data integration market.

The nature of operational data integration problems is different. Data integration is a small element of an application project unlike a data warehouse where DI may consume 80% of the project budget and timeline.

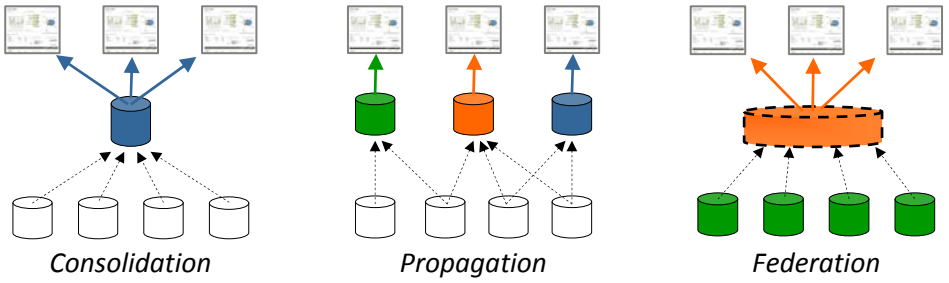
Most application integration projects need data from one or two other systems, not the many sources and tables feeding a data warehouse. The scope is usually smaller, with lower data volumes and narrower sets of data being transferred with minimal transformation.

A key challenge for OpDI is that the data is usually needed more frequently than one batch per night, unlike most analytic environments. Traditional ETL products for the data warehouse market don't handle low latency requirements as well as other integration tools. This makes ETL a poorer fit for some types of operational data integration.

The differences in frequency of execution, data volume, latency and scope are technical elements that differentiate operational and analytic data integration. The other characteristic that separates them is usage scenarios. How people integrate data in operational environments is different.

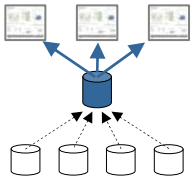
Three Approaches for Data Integration

The data integration scenarios commonly encountered in projects can be mapped to one of three underlying approaches: consolidation, propagation or federation. Consolidation implies moving data to a single central repository where it can be accessed. With propagation the data is copied from the sources to the application's local data store. Federation leaves data in place while centralizing the access mechanisms so the data appears to consuming applications as if it were consolidated.



Consolidation

The concept of consolidation is to move the data wholesale from one or more systems to another. All integration and transformation is done before it is loaded in the target system. This is most often seen in business intelligence, where ETL is used to centralize data from many systems into a single data warehouse or operational data store. Outside of analytic environments, a single centrally-accessed repository is most likely to be found in master data management and CRM projects.



Consolidation

In the world of operational data integration there are several other scenarios that fit within a consolidation approach. System migrations, upgrades and consolidations all require large-scale movement of data from one system to another.

Consider a merger or acquisition where there are redundant systems between the two companies. If the companies are running multiple instances of the same software they can reduce the cost of software maintenance and operations by consolidating these into one instance.

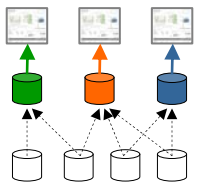
Merging the data from several instances of an ERP system is not a trivial task. There can be thousands of tables to copy and merge, and that's the simple part. Data quality issues are usually discovered in the process. The solution may require de-duplicating customer records, merging vendors, or reassigning and cross-referencing product numbers.

The advantage of not physically copying data means that there are no databases to create or tables to manage, speeding development.

The same tasks and problems occur in a single company when migrating from one vendor's application to another, for example when moving from an internal CRM system to a hosted application. Even packaged application upgrades can involve a level of data migration. Deploying new applications almost always involves importing data and setting up data feeds to and from other systems.

Propagation

Unlike the one time job of an upgrade, migration, or consolidation, propagation is an ongoing activity. Propagation is the most popular approach used for repetitive data integration because it's the simplest to implement. When an application needs data from another system, an automated program or database tool is used to copy the data. Data transformation, if any, is done as part of the process before loading the data into the target.



Propagation

Depending on the tools, propagation can be scheduled as a batch activity or triggered by events. Most of the time it is done as a push model from the source to the target, but it can also be implemented as a pull model driven by the application.

The data movement may be one-way or bidirectional. One-way data movement is common in scenarios where an application needs periodic data feeds or refreshes of reference data. For example, a product pricing system needs to send price updates to a web site, an order entry system and a customer service system.

Synchronizing data between systems is more challenging because it is bidirectional and can involve more than two systems. As the number of systems goes up, the number of possible connections explodes. Customer data is a common case where synchronization is used.

Many applications can touch customer data, for example order entry, accounts payable, CRM and SFA systems. Some changes, like credit status, customer contacts or refunds should be represented across all the systems when they occur. Because the systems are

independent, it isn't possible to centralize the data. Instead, the data needs to be synchronized so changes in one location are reflected in other locations.

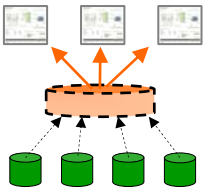
Propagation often leads to the need for synchronization because data is being copied and later changed in downstream systems. Data multiplies and discrepancies appear leading to disagreements about which information is correct.

Dealing with these problems at enterprise scale can be overwhelming because of the tangle of hand-coded integration that evolved over the years with the applications. Propagation is an easy and expedient solution without tools, but creates data management problems.

Data integration tools can help solve these problems. The common tool set and information collected in the tool metadata make it easier to understand and manage the flow of data. This in turn simplifies maintenance tasks and speeds both changes and new projects that require access to existing data.

Federation

Federation is a method for centralizing data without physically consolidating it first. This can be thought of as centrally mediated access or on-demand data integration. The data access and integration are defined as part of a model, and that model is invoked when an application requests the data.



Federation

Federated data appears to an application as if it were physically integrated in one place as a table, file or web service call. In the background a process accesses the source data in the remote systems, applies any required transformations and presents the results, much like a SQL query but without the restriction that all of the data originate in a relational database.

Because federation is a view imposed on top of external sources, it's generally a one-way flow of information. It can't be used to synchronize or migrate data between two systems. This makes federation appropriate for a different class of problems such as making data from multiple systems appear as if it came from a single source, or providing access to data that shouldn't be copied for security or privacy reasons.

Federation is a useful approach in scenarios where it would be too costly to create and manage a database for the integrated data. For example, in a customer self-service portal there might be a dozen possible sources of data the customer could access.

Pulling the required data from many systems into a single database is possible in this scenario. The challenge is providing real-time delivery of this information. A change in any of a dozen systems must be immediately replicated to this database – a challenging and expensive task. By federating access or constructing a data service layer ,the application developers can build the portal against a unified model without the need to copy data. The data is accessed directly from the source so there is no problem with delivering out of date or incorrect information.

Creating Solutions for Operational Data Integration Problems

Regardless of the data integration model, the final decision is usually governed by the project budget, timeline and what the developers are familiar with. IT focus on budget is at an all-time high making it hard to justify the investment needed for data integration tools. This is an area where open source can help.

The Most Common Practice: Custom Coding

Industry surveys show that operational data integration is built by hand for more than three quarters of the application projects in production today.

Products get better over time. Hand-written code gets worse.

Hand-coding is common because DI is not thought of in terms of infrastructure and data management, but in terms of glue for applications. While copying data from one place to another isn't optimal, it's still workable in the context of a single application. The price is paid in the overall complexity of integration spread throughout the enterprise.

Hand-coded integration is going to change due in large part to the new emphasis on external integration – for example with automated business processes involving outside companies or with the increasing use of SaaS applications.

Database administrators have no easy way to move data outside the company. The standard DBA tools do not allow them to send and receive data to the web service interfaces used by most SaaS applications, nor do DBAs have the expertise to program to these interfaces.

Application developers have the skills to send and receive remote data and to program to web services. The problem is that operational data integration is more than the core tasks of extracting and moving data. Reliable production support means creating components to deal gracefully with exceptions, handle errors, and tie in to scheduling, monitoring and notification systems. The additional work is enough to constitute its own project.

Migrations, upgrades and consolidations are a slightly different problem. The complexity and scale of mapping hundreds to thousands of tables makes the labor of hand-coding a poor choice. Beyond the amount of work, problems are hard to debug and there is no traceability for the data. The lack of easy traceability can create compliance and audit headaches after the new system is in production.

Hand-coding for operational data integration is a dead-end investment. Products usually improve over time. Extending this code or fixing minor problems is a low priority relative to other IT needs. Since the code is written for a specific project it can rarely be reused on other projects the way a tool can be reused.

The Standard Option: Buy a Data Integration Product

Integration code is single-purpose, tools are multi-purpose. You should always go with tools – when you can afford them.

Companies are recognizing the problems associated with hand-coded integration and are starting to evaluate and use data integration products. Coding requires proficiency with the operating system, data formats and language for every platform being accessed. DI tools improve productivity by abstracting work away from the underlying platforms. This allows the developer to focus on the logic rather than unimportant platform details.

There are a number of different tools available that can work for operational DI problems. Companies with a data warehouse are extending their use of ETL tools into this space. Tasks involving consolidation are particularly well suited to ETL tools because the problem domain matches their capabilities for large batch movement of data. The use is one-time so there is little danger of needing to pay for more licenses.

The large ETL vendors are shifting their product strategies to address operational DI needs and now call themselves data integration vendors. Their initial focus has been migrations and consolidations, although all have been reworking the tools to function better in propagation and synchronization scenarios where low latency data access is more important.

ETL tools are still a poor fit for propagation and synchronization because of their inability to address high concurrency, low latency needs. Other problems with many of the products are their complexity, deployment architecture, and cost.

Most are designed as centralized servers. This forces all integration jobs onto a single server or cluster which must then be shared with other users. It is possible to run smaller independent servers for different applications, but the cost of doing this is prohibitive because of the server-based licensing model.

Companies need tools that can be deployed in a distributed manner at the point of use, and that can be given to any application developer who needs them. Enterprise server licensing for ETL, DI and SOA tools often prevent this.

The Third Alternative: Open Source

Open source avoids the pitfalls of coding and gains the advantages of using tools.

Open source offers a third alternative to the traditional buy versus build decision. When looking for tools directed at developers, the first step should always be to look for open source software. Assuming there is an acceptable solution, it's clear that you will save time and money over custom development.

Given the three quarters of companies hand-coding integration, it's time to revisit the buy versus build decision. Open source data integration tools can address the shortcomings of hand-coding. As full featured tools, they offer the error handling, operational support and availability features that must be built in manual coding environments.

An advantage not often discussed is the productivity these tools bring to application developers. Aside from the standard integration tasks, they offer a significant improvement when dealing with heterogeneous systems and databases. DI tools expand the ability to do data integration to a developer audience who would otherwise lack the necessary platform skills.

Open source also has advantages over the current crop of DI tools on the market when it comes to operational data integration. The ability to expand traditional ETL tools for use in operational DI is limited because of the mismatch their centralized architecture and costly licensing models have with distributed OpDI needs.

The budget for application projects can't absorb the high cost of enterprise DI software which makes it hard to justify the purchase of a tool. Spending on infrastructure goes against project-based budgeting models and the ROI is hard to measure.

Data integration is, and will continue to be, viewed as application glue so organizations need an alternative. If IT can't afford to fund an enterprise DI tool as an infrastructure item then the alternative is either hand-coding or open source.

Open source data integration tools provide the cost advantages of hand-coding with the productivity advantages of traditional data integration software. This is the real reward for using open source development tools.

The Benefits of Open Source for Data Integration

People often misunderstand or misrepresent the benefits open source provides. The same could be said of packaged software in general.

The sad truth of most software is that it is non-differentiating. It does not confer any competitive benefit to the company because a competitor can acquire identical software. Likewise, data integration tools are themselves not a differentiator. The difference is that these tools allow developers the freedom to do better customized integration. They are an enabling technology that allows a company to differentiate how it configures systems and the flow of information. This is the point where differentiation occurs.

For this reason, most development tools have been taken over by open source. No company out-competes another by developing their own data integration software any more they would from building their own general ledger. With development tools, everyone wins by pooling their collective resources. The part they keep to themselves is what gets done with those tools because that's where the value is.

Data integration software is squarely in the crosshairs of open source vendors and venture capitalists because it fits the same profile as compilers, languages, and other development tools. In all of these cases the shared development and distribution model removed cost, improved tool quality, and benefited everyone.

There are two open source models for sharing development and distribution costs. One is project-based or community-based open source. The other is commercial open source software, or COSS for short.

Most people are familiar with project-based open source. This model typically involves some sort of non-profit foundation or corporation to own the copyright, and people contribute their efforts to development and maintenance. They may even be full-time employees, but the project does not operate in the same way a traditional software company does.

Commercial open source evolved with recognition that companies are willing to pay for support, service, and other less tangible items like indemnification or certifying interoperability. A commercial open source vendor operates just like a traditional software vendor, except that the source code is not shrouded in secrecy. This enables more and deeper interaction between the community of customers and developers, making the open source model more user-focused than the traditional model.

In contrast to the majority of projects, commercial open source vendors employ most of the core developers for their project and expect to make a profit while doing so. They provide the same services and support that traditional vendors do, and frequently with

more flexibility and lower cost. Some COSS vendors borrow elements of the proprietary vendors, like building non-open source add-on components or features that can be purchased in place of, or in addition to, the free open source version of the software.

The difference between COSS vendors and traditional vendors is as much about business practices as it is about the code. Proprietary vendors can't open the doors and invite bug fixes, design suggestions or feature additions, nor should they. The key force driving many open source projects is not innovative intellectual property, but the commodity nature of development software.

Studies on open source adoption have noted that other benefits can outweigh the cost advantages of open source. Not all projects are justified based on financial benefit. While advantages may be translated into financial terms, value can come from solving a particular problem sooner, enabling work that was previously not possible, or providing efficiency that allows people to be deployed to other tasks.

According to several market surveys over the past few years of companies adopting open source, the following three benefits rise to the top of the list.

Flexibility

The challenge with flexibility is defining it. Respondents use this term to mean a number of different elements of flexibility. These three are the most frequently mentioned:

Evaluation. Organizations can try open source tools at their own pace according to their own timeline. Some companies evaluate all tools in a proof of concept and allot the same amount of time to each. Others try open source first and run extended trials which evolve into prototypes or production use. Unlike traditional software, there are no non-disclosure agreements or trial licenses that limit the duration or extent of use, nor is there a presales consultant breathing down the neck of the evaluator.

Deployment. As noted above, a successful trial installation can be easily put into production. There are few, if any, limitations regarding deployment. For example, one firm may choose to centralize data integration while another chooses to distribute it closer to applications. Scaling up by adding servers is not usually limited with open source the way it is with traditional software models where more licenses must be purchased. The unbundling of license, support, and service mean decisions about these items can be made later.

Adaptability. Open source tools may be used in unrestricted ways, for purposes that the project might never have intended. Buying a six-figure ETL tool for a small integration problem or a one-time migration is overkill, but an open source ETL tool can be easily adapted for use. Another side of adaptability is customization. Most companies will rarely, if ever, look at a project's source code. It's still nice to know that the software can be tailored to fit a situation if the need arises – for example with the addition of customized connectors.

Vendor Independence

A benefit of open source mentioned by many customers is vendor independence. There are two aspects to vendor dependence. One is being beholden to a given vendor for the use and support of the software. The other is the problem of technology lock-in.

The open source license is the key difference for open source software. Even if a customer contracts with a COSS vendor in order to get support or other services, there is no requirement to continue with that vendor. This opens up the possibility of using third parties for the same services, or foregoing those services but continuing to use the software.

The problem of technology lock-in is much less likely to happen with open source software. Open source projects tend to adhere to open standards. There is more motivation to use existing open standards and reuse other open source code than to try to create new standards. The fact that the code is visible to everyone is an additional incentive to the developers to write better code. Studies have shown that many open source projects have lower defect rates than comparable proprietary offerings.

Proprietary vendors sometimes avoid open standards because proprietary standards ensure control over their working environment and the customer base. Some products are closely tied to vendor technology stacks, an obvious example being database-supplied ETL tools. Open source tools are much less likely to be tied to a specific platform or technology stack, partly because of how the software is developed and partly due to the diversity of the developer and user communities who quickly port useful code to their platform of choice.

Optimal Price

It's important to distinguish between cost savings and paying the right price. While the open source production and distribution model has cost advantages that translate directly into lower license price, this does not guarantee that a company will save money by using open source.

It's hard to justify even the lowest cost tools for a system migration because they become shelfware at the end of the project.

More important than trying to evaluate cost savings is looking at paying the right price at the right time. Open source gives a company the option to pay nothing, pay incrementally, or pay up front. The choice depends on factors like budget for initial project start-up, how important support is during development and anticipated growth once in production.

The greatest savings opportunities will come from new projects where the high cost of data integration tools favors open source. Start-up costs for a project using proprietary DI tools can be exceptionally high and you can't defer purchase or support costs with traditional software.

The next biggest savings comes when scaling for growth. As the number of servers, data source & targets, or CPUs grows, the license cost of proprietary tools keeps pace. Scaling up can quickly become cost-prohibitive.

Open source is delivered in ways that allow for low-cost or even zero-cost scaling. Some COSS vendors charge for support based on fixed attributes or simple subscription pricing. Others charge per developer rather than on a per-server basis.

For operational data integration, this translates into a significant advantage for open source. Most operational DI software is distributed across the enterprise, not in a few centralized servers. This poses serious cost obstacles for proprietary vendors.

The cost benefits of open source tools can be even higher for data consolidation tasks where the challenge is to justify the purchase of a tool that will be used once. Traditional enterprise data integration tools are not priced for one-time use, putting them out of reach for most projects.

It's hard for a manager to justify even the lowest cost tools because at the end of the project they become shelfware. For cases like this when expensive mainstream IT software is out of reach, open source can save the day.

Recommendations

The way organizations plan and budget for data integration is not going to change any time soon. Most operational data integration will continue to be paid for as part of individual projects, continuing the largely ad-hoc DI infrastructure. This means the single high-cost enterprise licensing model for new operational DI tools isn't likely to fit most IT organizations.

Operational DI is not the same as ETL or analytic DI. Keep this in mind when evaluating tools.

IT managers and developers need a way to make the integration job easier, repeatable and more productive. Open source is one way to accomplish these goals. People responsible for selecting and maintaining tools for data integration can benefit from the following guidelines.

- **Differentiate between analytic data integration and operational data integration.** Business intelligence environments have specific needs like large batch volumes, many-to-one consolidation and specialized table constructs. While applicable to consolidation projects, ETL tools designed for the data warehouse market won't provide a complete set of features for operational data integration.

- **Discourage hand-coded data integration.** There are many different tools which can be used to solve data integration problems, and newer tools specifically designed for operational data integration. Encourage developers on application development and package implementation projects to look at these tools. The benefits over manual coding are obvious.

- **Use the right data integration model for the problem.** Determine whether the integration problem requires consolidation, federation or propagation. Each of these is different in both approach and required tools or features. Select the technology that best fits with the approach to avoid mismatches that will lead to problems during implementation.

- **Make open source the default option for data integration tools.** When in an environment with few or no tools, open source should be the first alternative. It is the simplest, fastest and likely the least expensive route to solve the problem. It's the logical next step after manual coding. Look to proprietary tools only when open source tools can't do the job, or when you have them in house already and the licensing issues are not an obstruction.

- **Augment existing data integration infrastructure with open source.** There will be many cases where it is not effective to extend current data integration tools to a new project. This may be due to lack of specific features, poor fit with the application architecture, or extended cost due to licensing or the need for additional components. Many proprietary data integration tools will charge extra for options like application connectors, data profiling or data cleansing. In these cases, open source can be used to augment the existing infrastructure.

About the Author

MARK MADSEN is president of Third Nature, a consulting and technology research firm focused on information management. Mark is an award-winning architect and former CTO whose work has been featured in numerous industry publications. He is an international speaker, a contributing editor at Intelligent Enterprise, and manages the open source channel at the Business Intelligence Network. For more information or to contact Mark, visit http://ThirdNature.net.

About the Sponsor

Talend is the recognized market leader in open source data integration. Hundreds of paying customers around the globe use the Talend Integration Suite of products and services to optimize the costs of data integration, ETL and data quality. With over 3.3 million lifetime downloads and 700,000 core product downloads, Talend's solutions are the most widely used and deployed data integration solutions in the world. The company has major offices in North America, Europe and Asia, and a global network of technical and services partners. For more information and to download Talend's products, please visit http://www.talend.com.

About Third Nature

Third Nature is a research and consulting firm focused on new practices and emerging technology for business intelligence, data integration and information management.

Our goal is to help companies learn how to take advantage of new information-driven management practices and applications. We offer consulting, education and research services to support business and IT organizations as well as technology vendors.