FOREWORD

NUMBER ONE
Enable real-time data warehousing with real-time data integration.

NUMBER TWO
Know the available real-time data integration techniques.

NUMBER THREE
Virtualize and pool data resources for access in real time or other speeds and frequencies.

NUMBER FOUR
Profile, certify, improve, and optimize data as you virtualize it.

NUMBER FIVE
Abstract real-time data warehouse functionality as data services.

NUMBER SIX
Provision data services for any application, in multiple modes and protocols.

NUMBER SEVEN
Leverage a single platform for both physical and virtual data integration.

ABOUT OUR SPONSOR

ABOUT THE AUTHOR

ABOUT TDWI RESEARCH
In a 2009 TDWI survey, a paltry 17% of survey respondents reported using real-time functionality with their data warehouses. Yet a whopping 92% said they would be using it within three years. This puts real-time functions at the top of the priority list for next-generation data warehouses, just ahead of other much-needed functions such as master data management, cloud computing, and analytics.

Why the rush to real-time data warehousing? Because it’s a foundation for time-sensitive business practices that users are adopting today. Some of these practices are purely a matter of timely decision making, such as operational business intelligence and frequently refreshed management dashboards. Others support fast-paced operations, such as just-in-time inventory, facility monitoring, and on-the-fly recommendations in e-commerce.

What is real-time data warehousing (RT DW)? It’s a combination of speedy technologies and fast-paced business processes. The technology handles real-time data transfers or federated data fetches from operational applications into a data warehouse and beyond into reports and dashboards. The point is to give business managers the information they need—and an appropriate presentation of it via business intelligence (BI) platforms—for making time-sensitive decisions relative to daily operations.

What is the best way to implement RT DW? This TDWI Checklist Report asserts that real-time data warehousing is best enabled by real-time data integration tools and techniques. This is because modern DI platforms and tools support a number of techniques that are conducive to RT, such as microbatches, data federation, real-time data quality, changed data capture, complex event processing, and so on.

How does data integration support RT DW? To satisfy diverse business requirements for information delivery, RT DW needs multiple RT DI techniques that can operate at multiple speeds and frequencies and handle both physical (persistent) data and virtual (instantiated) data. This report lays out a methodology for virtualizing enterprise data, so that data is easily provisioned via data services that can operate in real time, latently in bulk, or any gradation between.

If you’re facing a RT DW implementation or other work involving RT DI, you should consider the recommendations of this TDWI Checklist Report.
Most of the real-time functionality applied in data warehousing and business intelligence is enabled by various types of data integration tools and techniques, along with the diverse data delivery speeds and interfaces they support. After all, these DI techniques fetch data and integrate it into the DW in real time. Yet, we still use the term real-time data warehousing, as well as synonyms such as active data warehousing, dynamic data warehousing, and operational data warehousing.

Accommodating real-time information delivery usually involves adjustments to data warehouse architecture and data models, as well as how reports are refreshed:

**Staging area for incoming real-time data.** A data warehouse architecture may need a data staging area to accommodate data arriving in real time. There’s a long-standing tradition of modeling an operational data store (ODS), which may be an integral part of the DW architecture (as a collection of tables within the main DW) or deployed as a standalone system. Either way, an RT ODS can accept real-time data as transactions or frequent batches, then process the data into forms that the DW can load and handle.

**Responsive queries to pull real-time data back out of the warehouse.** Accelerating query response may be accomplished via indices, materialized views, columnar data storage, or in-memory data management.

**Reactive interfaces to push real-time data to another platform.** Although real-time data flow is mostly driven by a data integration platform, the DW platform’s database management system (DBMS) may also need to support interfaces that monitor data values and react when they enter prescribed parameters. This may entail DBMS functions such as messages, events, triggers, real-time alerts, Web services, and so on. Through these, the DW can push data to another platform (such as an operational application) or trigger an event on another system (such as a report refresh on a BI platform).

People who are new to RT DW commonly ask: “Do I really need a real-time data integration tool and a real-time data warehouse platform when all I want to do is move data from an operational application into a report in real time?” This is a reasonable question, because all good BI platforms support distributed queries and data federation, and these can fetch operational data and move it directly into a report in real time. However, this is a quick and dirty process; the data delivered this way is in relatively poor condition compared to data that results from data integration and data warehousing. (See Figure 1.)

**Data integration is critical to transforming and improving real-time data.** DI transforms complex data into useful and reusable forms. It improves data’s quality, metadata, and master data. When done well, DI assures that data complies with policies for data access, use, security, privacy, and data standards.

**A data warehouse is critical to forming a complete and historic context for real-time data.** A DW can provide additional data for a complete historic context that complements point-in-time, real-time data. Even so, if real-time data is useful without contextual data from a warehouse, a data integration service may send transformed and cleansed data directly to a BI platform or other target.

Hence, circumventing data integration and data warehouse infrastructure when handling real-time data puts the data and the business processes that rely on it in peril. Indeed, it’s possible to get current data by going around DI and DW. But the data won’t be fully repurposed, clean, compliant, complete, contextual, or auditable.

---

1 For an in-depth discussion of real-time data warehousing and its variations, see the TDWI Best Practices Report Operational Data Warehousing, available online at tdwi.org/research/list/tdwi-best-practices-reports.

---

**Figure 1.** Most real-time data for BI should travel through data integration and data warehousing infrastructure.
KNOW THE AVAILABLE REAL-TIME DATA INTEGRATION TECHNIQUES.

Note that the term *real time* doesn’t necessarily mean “instantaneous.” Real time is actually a blanket term that encompasses a range of information delivery speeds and frequencies, which are triggered through scheduling or on-demand requests.

**Real time—literally.** For example, a monitoring application for managing an electrical utility needs data delivered literally in milliseconds, so that problems can be addressed by human action within seconds or minutes to keep the grid performing optimally. In this example, accessing, processing, and delivering data must execute in milliseconds. This may require special technologies for event processing, message buses, or services. Note that this example is about the *speed* of execution of a DI solution.

**Near time.** Few business processes can respond so quickly. For most, data can arrive more leisurely in near time. For example, a just-in-time inventory application deals with time-sensitive information where human actions can still be effective if a response is delayed for a few hours. Data can travel more slowly, perhaps through batch processing over traditional interfaces. Note that this example is not about speed of execution. It’s about the *frequency* of execution of a DI solution.

**Frequent intraday batches.** One of the many characteristics of traditional solutions for ETL is that they are scheduled. That’s so the ETL solution hits source and target systems in off-peak periods, usually in the middle of the night. This is fine, since most BI methods don’t need data fresher than 24 hours old. But operational BI and management dashboards are an exception. Both rely on very fresh operational data, which means that managers need to refresh their reports and dashboards frequently. These practices are driving some DWs into frequent intraday microbatches that refresh certain time-sensitive data (not the whole warehouse).

**On demand, any time.** Some managers need to refresh reports and dashboards on demand, any time they want. Instead of being scheduled, on-demand DI (and the report refresh that ensues) executes unpredictably. Some managers get reports refreshed on demand in milliseconds. Others have to wait a few minutes, but that’s far better than having to wait for the next scheduled batch update.

As these examples illustrate, the business requirements for the speed and frequency of RT DI solutions vary tremendously. Satisfying the diverse requirements requires a portfolio of diverse data integration techniques and tool types, which can operate at various speeds, frequencies, and other performance characteristics, as well as through diverse interfaces and protocols.

Luckily for data integration specialists, good DI tools support a number of DI techniques that are conducive to real-time operation. These include data federation, microbatch processing, changed data capture (CDC), complex event processing (CEP), on-the-fly (or real-time) data quality, Web services, service-oriented architecture (SOA), and interoperability with message and service buses. That’s a long and varied list, because, again, the business requirements for information delivery through a real-time data warehouse and other systems are equally diverse.

**Use Cases for Data Federation**

Based on distributed query technology, but extended through services, data federation is capable of fetching operational data and refreshing a report with that data in real time. As pointed out earlier, however, data federation has limited data processing capabilities compared to DI and DW solutions. Yet, its speed of execution has made it a prominent DI technique in recent years. Here are a few use cases that prove the usefulness of data federation:

- **Prototype before you persist.** Once errant data is loaded into a DW, it’s hard to get out again. Use federation to design and test reports before committing their data to a DW.
- **Augment DW with current data.** Most DWs keep a history of corporate performance, whereas federation provides a snapshot. Combine these sources for a complete view.
- **Federate DWs and marts.** Federated data marts are based on data from a DW. Instead of moving data to the marts, access it via federation.
- **Augment ETL.** Federation gathers current data in small amounts with simple transforms. ETL’s data is latent, large, and complex. The two work well in tandem.
- **Data services layer in SOA.** Some data federation tools support Web services and SOA, which make the tool an entry point for integrating data across a service architecture or bus.
For the purposes of this Checklist Report, let’s define data virtualization as the pooling of data integration resources. This is similar to how the pooling of server resources creates a virtualized cloud for applications, from which server resources are freely allocated, freed up, and reallocated as applications need them. But data virtualization pools data, as seen through the data sources, targets, transformations, interfaces, protocols, and other components of a data integration solution. From the pool of data integration resources, data can be provisioned freely, as applications demand it.

Data virtualization via DI techniques has benefits:

**Data virtualization provides a business-friendly abstraction layer.** This layer insulates DI targets from changes in DI sources as new data sources are added or retired (e.g., through a merger or data mart consolidation). The layer also enables data provisioning—even in real time. Real-time provisioning is key, because the kind of data you want to move in real time for BI/DW changes so rapidly that only the most recent update is useful for a business process. Most important, when this layer exposes data objects for business entities (such as customer, order, and so on), it provides a business-friendly view that abstracts yet handles the underlying complexity.

**Data virtualization fosters collaboration.** It can promote efficient and effective collaboration between business and IT personnel when built on top of a single metadata repository with role-based tools and a solid data integration platform supporting all styles of data integration.

**Data virtualization ensures trusted data.** It can ensure that inconsistencies and inaccuracies across heterogeneous enterprise data stores are identified up front without the need for staging and pre-processing. In addition, it enables business and IT to collaborate in defining and enforcing data quality rules on the fly or in real time as data is federated.

**Data virtualization enables reusable objects and services.** It allows you to design data object models and services once, federate access across fragmented or siloed data sources, and provision the data however it is needed by consuming channels and applications (e.g., SQL, Web services, message queues, batch ETL, and so on).

**Data virtualization supports real-time data warehousing.** The virtualized pool of data integration resources can provision data to populate any target, including data warehouses. Since virtualization means that most DI objects and services are capable of operating in real time—as well as other speeds and frequencies—real-time data for the data warehouse is likewise available from the pool.

**Data Virtualization and Traditional Data Federation Compared**
Data virtualization and data federation are closely related, which may be why so many people confuse the two or think they are identical. As the term suggests, data virtualization must abstract the underlying complexity and provide a business-friendly view of trusted data on demand. To avoid confusion, it’s best to think of data federation as a subset or component of data virtualization. In that context, you can see that a traditional approach to federation is somewhat basic or simple compared to the greater functionality of data virtualization. To sort out the differences, let’s compare traditional data federation and data virtualization. (See Table 1, next page.)

- As data is pooled via data virtualization, it may be physically relocated and persisted or accessed on the fly like data federation.
- Data transformations and table joins are relatively simple with data federation, whereas data virtualization offers the full gamut of rich, ETL-style complex transformations.
- Data federation is limited to relatively small data sets, but data virtualization scales up large ones.
- Data virtualization supports the full best practice of improving data and certifying it for an intended use, while data federation simply grabs and hands off data without much processing.
- Data federation capabilities should be built into a comprehensive DI platform so that they are available for data virtualization alongside other, complementary DI techniques.
- The nature of data virtualization is to promote multi-use DI objects and data services that can apply given DI logic via multiple DI techniques, such as data federation, ETL, data sync, CDC, and so on.

(Continues)
(Continued)

Table 1

<table>
<thead>
<tr>
<th>TRADITIONAL DATA FEDERATION</th>
<th>DATA VIRTUALIZATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Only federates data from many different data sources in real time</td>
<td>Processes and delivers in any way (physical/virtual data integration)</td>
</tr>
<tr>
<td>Limited support (SQL/XQuery only) for complex transformations</td>
<td>Full range of rich, ETL-style complex transformations</td>
</tr>
<tr>
<td>Does not handle large data sets (e.g., millions of rows in one query)</td>
<td>Single environment that supports bulk processing and federation</td>
</tr>
<tr>
<td>Does not certify the quality of data—simply propagates bad data in real time</td>
<td>Proactively identifies and fixes data quality issues on the fly in the same tool</td>
</tr>
<tr>
<td>Causes delays due to lack of collaboration between business and IT</td>
<td>Promotes iterative collaboration and validation between business and IT</td>
</tr>
<tr>
<td>Requires separate tools, skills, and re-training for ETL and data federation</td>
<td>Leverages same environment and skills for batch ETL and data federation</td>
</tr>
<tr>
<td>Requires rebuilding or redeployment of the same data integration logic for batch ETL</td>
<td>No rework—no rebuilding or redeployment of data integration logic</td>
</tr>
</tbody>
</table>

Table 1. Data virtualization is not traditional federation. (Source: Informatica)

Data integration in support of data warehousing must adhere to a series of data management best practices if the data entering the data warehouse is to be in the best possible condition for use across multiple reports, analyses, and other BI solutions. Note that the practices described below apply to DI in the context of data virtualization, not just traditional DI methods.

**Profiling.** Data virtualization takes advantage of integrated data profiling to enable column profiling and join analysis to ensure the success of the resulting federation. Typically, this is not handled by data federation and requires staging and pre-processing the data.

**Certification.** As data and DI logic are admitted into the virtualized pool, they are certified up front for data quality, security policies, compliance regulations, data standards, and so on. This makes the pool a trusted source, anoints it as the single source that most BI applications should use, and accelerates the provisioning of data later.

**Data quality (DQ).** Assuring the quality of data per established metrics is an assumed responsibility of data virtualization. By comparison, a standalone data federation tool rarely provides any form of data quality improvement. Or, it may require a separate tool and skills to cleanse the data after the data federation process, thus causing project delays and defeating the purpose of having real-time data federation. A data virtualization tool that leverages a full-featured DI platform can provide the full range of data profiling and data quality functions, and even execute them in real time. For example, DQ functions for data standardization, verification, validation, data append, ID resolution, and localization are especially effective in real time.

**Metadata and master data.** Documenting data’s use, meaning, origins, and so on via extended metadata and master data are improvements that make data objects and DI services in the virtualized pool more reusable and auditable, plus better suited to review and maintenance by a wide range of personnel.

**Performance optimization.** As DI objects and services are developed, they should be optimized for performance. Partitioning, grid computing, pushdown optimization, caching optimization, and so on are examples of runtime deployment capabilities that improve performance and scalability.
There’s a problem with the development process many organizations are applying to RT DI and RT DW. They developed a rich library of data integration routines, logic, and interfaces for their high-latency, history-oriented data warehouses. Then they developed a second library of equivalent DI functions, but for operation in real time, intraday, or on demand. (See Figure 2a.) There may be still more libraries for classes of interfaces and individual projects.

Redundancy is the real problem here. Both libraries touch the same data, load the same DW, use the same or similar DI logic, and so on. Redundant libraries result in problems with complexity, maintenance, developer productivity, contradictory DI logic, consistent data use, compliance, governance, data standards, and more.

Instead of this inefficient redundancy, a better approach is to have one library that can operate flexibly. Instead of a project-based development method, take a model-driven approach where you build assets for reuse across projects or repurposing within projects. A single library contains non-redundant data services that are reusable across a number of workloads, interfaces, and projects. (See Figure 2b.)

Many benefits ensue. Data services are handy for embedding data and DI logic in multiple application types, including composite applications. As developers reuse data services, they also reuse the policies and business rules built into them for greater compliance and governance. Over time, the reduction of DI assets resulting from a data services approach can lead to less development and maintenance effort.

However, this approach assumes that most data services are capable of operating flexibly in multiple “modes.” As explained in the next section of this report, a truly virtualized data service operates in real time, latently, or a gradation of these, as evolving technology and business requirements demand.
Achieving the broad reuse of data integration services and the single-library benefits mentioned earlier requires services that can operate in multiple modes. Multi-modality also makes your data services resilient to change as data volumes grow, latency requirements change, new data sources are added, and new applications or upgrades are provisioned with data.

A mode or modality can be a number of things:

**DI techniques invoked.** These depend on source and target types and the desired information delivery speed. DI techniques range from traditional batch data processing and CDC to event processing and messaging.

**DI logic applied.** In particular, the hard work put into developing complex data transformations should be leveraged across multiple uses of a data integration service, as well as across numerous DI projects.

**Interfaces and protocols a DI service may employ.** It’s important to have connectors and adapters that utilize native APIs wherever possible and are tightly integrated with the DI processing pipeline for optimal performance. These range from traditional ODBC/JDBC and the proprietary APIs of packaged applications to modern Web services and SOA.

**Multi-modal data services offer several benefits.** Multi-modal data services enable a kind of “any-speed data warehouse,” not just a real-time data warehouse. Supporting multiple modes in a single data service eliminates redundant DI libraries. Multi-modality results in fewer services designed and built, which promotes greater consistency for development and data standards, plus simpler policy compliance for data governance, security, and privacy. To achieve these benefits, you need a data integration tool that allows you to focus on the logic of a data integration service as you design it, then decide its mode each time you deploy it for runtime.

The modern methodology for designing and deploying data integration solutions described here—based on single-library, multi-modal data services—works best when built atop a single platform for data integration, as opposed to integrating multiple tools from diverse vendors. For example, achieving a single library of DI services for all modalities is far simpler with a single platform, as are other benefits such as reuse, compliance, and consistent standards. An assumption is that the single platform is capable of all the numerous DI techniques and modalities discussed here, all in a unified architecture and deployment package.

Relative to real-time data warehousing, one of the most useful benefits of a single platform is that it promotes the seamless reuse of data services between two essential modalities: physical DI and virtual DI. In other words, development artifacts such as transformation logic, data quality measures, and master data rules can be reused and leveraged in different data access and movement scenarios, whether the mode is physical (relocating data latently in bulk) or federated/virtual (fetching comparatively smaller data sets as needed). This combination of complementary modalities is very powerful because it processes and delivers information in any way that an application or user needs it, whether the required information is physical (persisted), virtual (instantiated), or both.

This combination of modalities is especially powerful for real-time data warehousing. As user organizations press deeper into time-sensitive decision making—such as operational business intelligence and management dashboards—they need ever-changing operational data delivered in real time, near time, and on demand. Data virtualization approaches to data integration enable that. But users also need a context for real-time data, which the slowly changing historic data of a physical data warehouse provides. In many cases, it’s the same or very similar information, whether physical or virtual; it’s just been captured in different states at different times. Therefore, the reuse of DI logic via different protocols and across these two modalities (physical and virtual) is key to assembling complementary data, but doing so with reuse, consistency, and compliance.
Informatica is the world’s number one independent provider of data integration software. Thousands of organizations rely on the Informatica Platform to obtain timely and trusted data for their most important business initiatives. With Informatica, enterprises gain a competitive advantage from all their information assets to grow revenues, increase profitability, further regulatory compliance, and foster customer loyalty. Informatica delivers a comprehensive, unified, open, and economical platform to reduce IT costs and expedite the time to address data integration needs of any complexity and scale.

www.informatica.com

TDWI Checklist Reports provide an overview of success factors for specific projects in business intelligence, data warehousing, or related data management disciplines. Companies may use this overview to get organized before beginning a project or to identify goals and areas of improvement for current projects.

Philip Russom is the senior manager of TDWI Research at The Data Warehousing Institute (TDWI), where he oversees many of TDWI’s research-oriented publications, services, and events. He’s been an industry analyst at Forrester Research, Giga Information Group, and Hurwitz Group, where he researched, wrote, spoke, and consulted about BI issues. Before that, Russom worked in technical and marketing positions for various database vendors. You can reach him at prussom@tdwi.org.

TDWI Research provides research and advice for BI professionals worldwide. TDWI Research focuses exclusively on BI/DW issues and teams up with industry practitioners to deliver both broad and deep understanding of the business and technical issues surrounding the deployment of business intelligence and data warehousing solutions. TDWI Research offers reports, commentary, and inquiry services via a worldwide Membership program and provides custom research, benchmarking, and strategic planning services to user and vendor organizations.