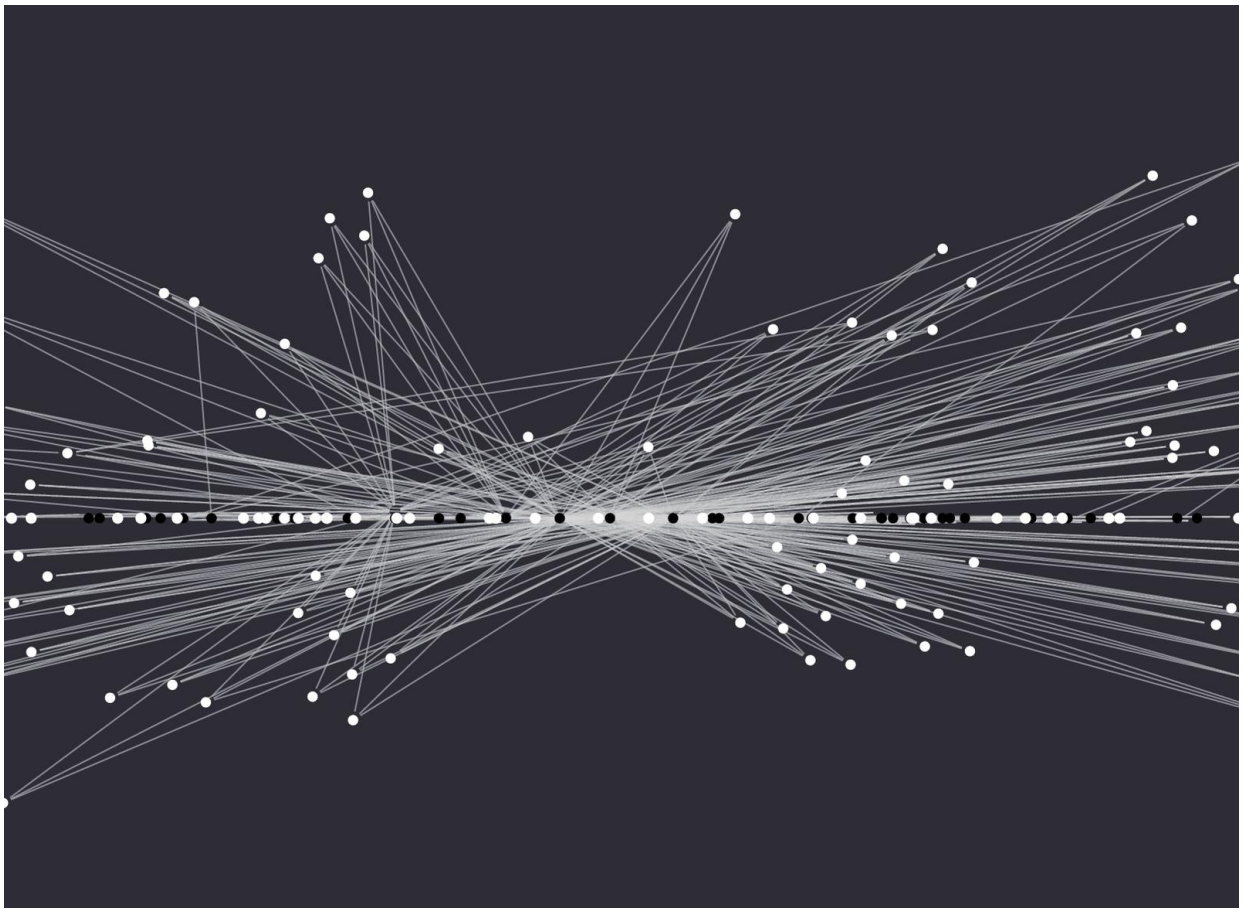
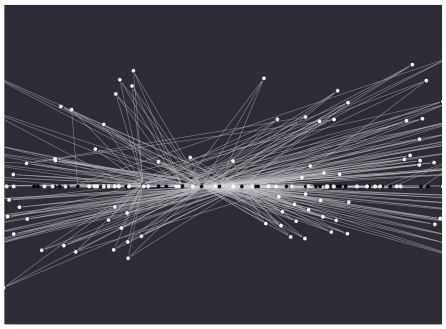
The Evolution of Real-Time Business Intelligence and How to Achieve It Using HPE Shadowbase Software

A Gravic, Inc. White Paper



Executive Summary



In today's competitive environment with high consumer expectation, business decisions based on the most current data available to the enterprise are necessary to improve customer relationships, increase revenue, and maximize operational efficiencies.

The speed of today's processing systems has moved classical data warehousing into the realm of real-time. The result is *real-time business intelligence* (RTBI). Business transactions are fed as they occur to a RTBI system that maintains the current state of the enterprise. The RTBI system not only supports the classical strategic functions of data warehousing for deriving information and knowledge from past enterprise activity, but it also provides real-time tactical support to drive enterprise actions that react to immediate events. As such, it replaces (or enhances) both the classical data warehouse and the *enterprise application integration* (EAI) functions. An RTBI system with a consolidated real-time view of the business across the enterprise enables the provision of new business services (applications) which were previously impossible, achieving competitive advantage.

RTBI is also known as *operational business intelligence* (OBI) and *event-driven business intelligence (EDBI).* In order to react in real-time, a business intelligence system must react to events as they occur – not minutes or hours later. With RTBI, an enterprise can establish long-term strategies to optimize its operations and maximize competitiveness while at the same time reacting with intelligence to events as they occur.

In this white paper, business intelligence is tracked from its early days as a strategic tool to today's real-time capabilities for tactically responding to events as they happen. In conclusion, descriptions of the HPE Shadowbase suite of products (built by Gravic, sold by HPE) are described, as well as several case studies where the products are deployed. HPE Shadowbase solutions are available today to help you achieve the benefits of RTBI.

Table of Contents

Table of Figures

The Evolution of Real-Time Business Intelligence and How to Achieve it Using HPE Shadowbase Software

Introduction

A decade ago, it was enough for a store to remain competitive if it could determine that battery-operated power drills were hot items and were sufficiently stocked. That ability is no longer adequate in the current aggressive marketing environment. Today, the same store needs to know that a customer who is purchasing a set of drill bits bought a battery-operated power drill the prior month. While the customer is at the register, the store offers him a 30% discount on an extra drill battery for purchase during his next visit. Similarly, the store offers an online customer with a similar purchasing history the same 30% discount during his current shopping session.

This requirement for current information (so called “fast data”) is needed across the board, and even aids in life-saving and rescue operations. A decade ago, a blood bank was considered successful in meeting the needs of a community if it could determine the most common blood types among the residents and could maintain sufficient supplies in the local distribution center. Now, that same blood bank is expected to supply all blood types, not just the common ones. If a local resident with a rare blood type is in a serious car accident, the blood bank either has to have that blood type stored locally or know where it can be obtained on a moment’s notice.

In today’s competitive environment of high consumer expectations, decisions that are based on the most current data available improve customer relationships, increase revenue, maximize operational efficiencies, and even save lives. This technology is real-time business intelligence (RTBI), also known as operational business intelligence (OBI) and event-driven business intelligence (EDBI). RTBI systems provide the information necessary to strategically improve an enterprise’s processes as well as to take tactical advantage of events as they occur.

In this white paper, business intelligence is tracked from its early days as a strategic tool to today’s real-time capabilities for tactically responding to events as they happen. In conclusion, descriptions of the HPE Shadowbase suite of products (built by Gravic, sold by HPE) are described, as well as several case studies where the products are deployed. HPE Shadowbase solutions are available today to help you achieve the benefits of RTBI.

Business Intelligence

Business intelligence is the use of an organization’s disparate and far-flung data to provide meaningful information and analyses to employees, customers, suppliers, and partners for more efficient and effective decision-making. It transforms information into actionable strategies and tactics to improve the efficiency of the enterprise, to reduce costs, to attract and retain customers, to improve sales, and to provide many other significant benefits.

As information technology evolved over the years, enterprises automated more and more of their operations. A great deal of valuable data resided underutilized in these systems. Data found in sales, accounting, production, human resources, and many other systems could yield significant information to provide historical, current, and predictive views of business operations.

For example, some typical instances of the use of business intelligence include:

Retail: Sales Patterns Integrated Customer View Campaign Management Customer Valuation Analytical CRM	**Manufacturing:** Order Life Cycle Inventory Analysis Quality Assurance Supplier Compliance Distribution Analysis	**Government:** National Security Crime Analysis Health Welfare Fraud Detection
Telecom: Call-Behavior Analysis Fraud Detection Number Portability Service-Usage Analysis Promotion Effectiveness	**Financial:** Credit Risk Monetary Risk Asset Management Liability Management Fraud Detection	**All Industries:** P & L Analysis Profitability Performance Analysis Value-Chain Analysis Profiling

The History of Business Intelligence

The Early Days of Computing

Typically, early business applications had their own, private databases that supported their functions, which became silo'd because no other systems had access to them. These *islands of information* proliferated as more and more departments were automated. Mergers and acquisitions compounded the problem since the companies integrated totally different systems, many of which were doing the same functional jobs.

Businesses soon recognized the analytical value of the data that they had available in their many islands of information. In fact, as businesses automated more systems, more data became available. Collecting this data for analysis was a frustrating challenge due to the incompatibilities between systems because there was no simple way (and often no way) for these systems to interact. An infrastructure was needed for data exchange, collection, and analysis that could provide a unified view of an enterprise's data, so the *data warehouse* and later, the *operational data store* (ODS), evolved to fill this need.

The Data Warehouse

Figure 1 depicts the online data warehouse concept: a single system for the repository of all of an organization's operational data. For example, gift registry, sales promotions, inventory, and point-of-sale applications feed data into the data warehouse via extract, transform, and load) batch transfers. Knowledge workers and management have a dashboard for pertinent information, and can run ad-hoc queries on the data warehouse, generating meaningful reports for strategic decision-making.

However, meeting the goal of a single repository for all of an organization's data presents several very significant challenges:

- **Data must be acquired** from a variety of incompatible systems.
- **The same item** of information might reside in the databases of different systems in different forms. A particular data item might not only be represented in different formats, but the values of this data item might be different in different databases. Which value is the correct (or current) one to use?
- **Data is continually changing**. How often should the data warehouse be updated to reflect a reasonably current view?
- **The amount of data is massive**. How is it analyzed and presented simply so that it is useful?

To meet these needs, a broad range of powerful tools were developed over the years and became productized. They included:

- **Extract, transform, and load (ETL) utilities** for the moving of data from the various data sources to the common data warehouse.
- **Data-mining engines for complex predetermined analyses** and ad hoc queries of the enterprise data stored in the data warehouse.
- **Reporting tools to provide management and knowledge workers** with the results of the analyses in easy-to-absorb formats; digital dashboards are a predominant example.

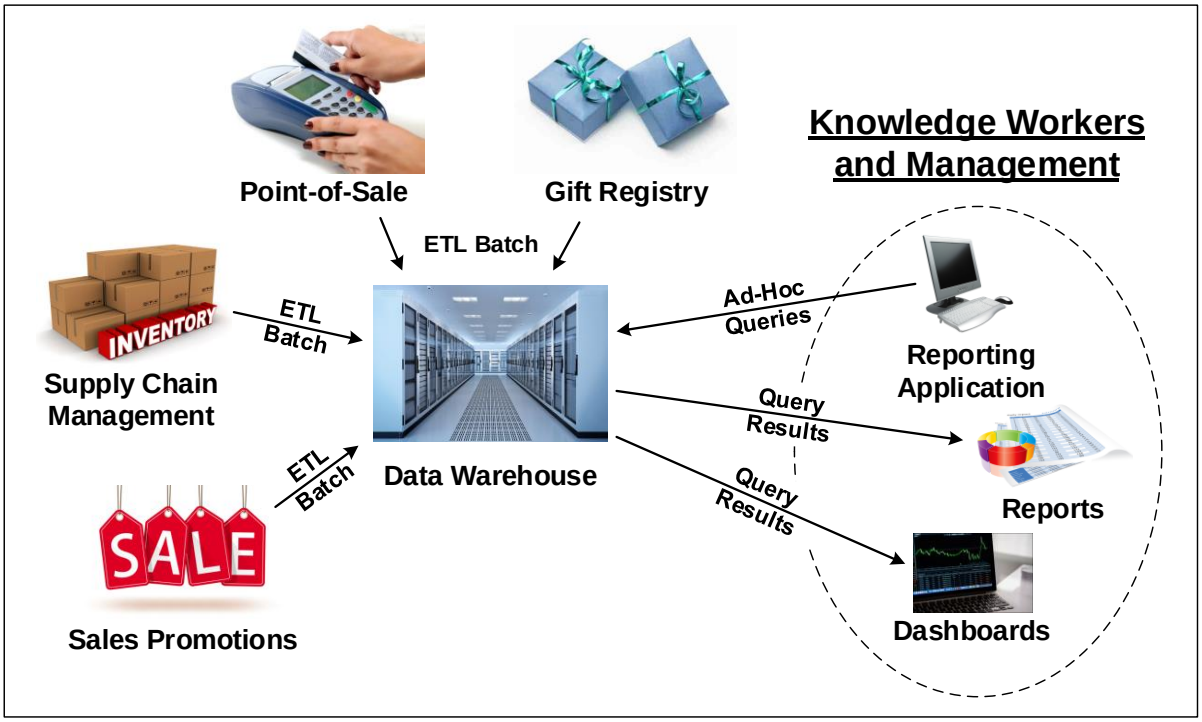


Figure 1 – The Data Warehouse

Offline Extract, Transform, and Load

Early on, the one common interface that was provided between the disparate systems in an organization was magnetic tape. Tape formats were standardized, and any system could write tapes that could be read by other systems, and so the first data warehouses were fed by magnetic tapes prepared by the various systems within the organization. However, that left the problem of data disparity, since the data written by the different systems reflected their native data organizations. The data written to tape by one system often bore little relation to similar data written by another system.

Even more important was that the data warehouse's database was designed to support the analytical functions required for the business intelligence function. This database design was typically a highly structured database with complex indices to support online analytical processing (OLAP), e.g., in an OLAP cube. Databases configured for OLAP allowed complex analytical and ad hoc queries with rapid execution time. The data fed to the data warehouse from the enterprise systems was converted to a format meaningful to the data warehouse. To solve the problem of initially loading this data into a data warehouse, keeping it updated, and resolving discrepancies, *extract, transform, and load* (ETL) utilities were developed.

Figure 2 shows the general flow of information (represented by the arrows) into a data warehouse using ETL utilities. First, the data from a legacy system database is extracted and placed in an intermediate database. The data is then transformed into a common data warehouse format and properly loaded into the warehouse database. This process solves the problem of data format disparity between systems.

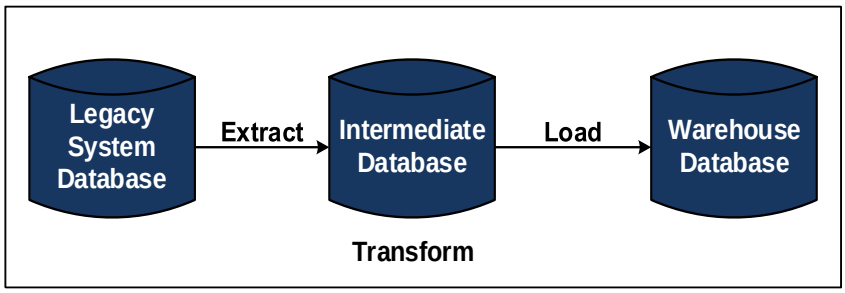


Figure 2 – ETL: Extract/Transform/Load

As their name implies, these utilities *extract* data from source databases, *transform* them into the common data warehouse format (the intermediate database), and *load* them into the warehouse database. The transform function is key to the success of this approach. It applies a series of rules to extracted data so that the data is properly formatted for loading into the data warehouse. Examples of transformation rules include:

- The selection of data to load
- The translation of encoded items (for instance, 1 for male, 2 for female to M, F)
- Encoding and standardizing free-form values (New Jersey, N. Jersey, N. J. to NJ)
- Deriving new calculated values (sale price = price – discount)
- Merging data from multiple sources
- Summarizing (aggregating) certain rows and columns
- Splitting a column into multiple columns (for instance, a comma-separated list)
- Resolving discrepancies between similar data items
- Validating the data
- Ensuring data consistency

The ETL function allows the consolidation of multiple data sources into a well-structured database that is used for complex analyses. The ETL process is executed periodically, such as daily, weekly, or monthly, depending upon the business needs. This process is called *offline ETL* because the target (warehouse) database is not continuously updated; it is updated on a periodic batch basis.

Though offline ETL serves its purpose well, it has some serious drawbacks. First, the data in the data warehouse is stale and could be days or weeks old, making it useful for strategic functions but not particularly adaptable to tactical uses. Second, the source database typically must be quiesced during the extract process; otherwise, the target database is in an inconsistent state following the load. As a result, the applications must be shut down, often for hours, resulting in loss of business services to customers.

In order to evolve to support RTBI, the ETL function must be continuous and noninvasive. This function is *online ETL*, which is described later. In contrast to offline ETL, which provides stale but consistent responses to queries, online ETL provides current but varying responses to successive queries since the data that it is using is continually being updated to reflect the current state of the enterprise.
Offline ETL technology has served businesses for decades and the intelligence that is derived from this data informs long-term reactive strategic decision-making. However, short-term operational and proactive tactical decision-making continues to rely on intuition, and the business service outage associated with offline ETL is unacceptable in today's 24-hour global economy. Data-Mining Engines

- The ETL utilities make data collection from many diverse systems practical. However, short-term operational and proactive tactical decision-making continues to rely on intuition, and the business service outage associated with offline ETL is unacceptable in today's 24-hour global economy. The captured data needs to be converted into information and knowledge in order to be useful: data are simply facts, numbers, and text that can be processed by a computer. For instance, a transaction at a retail point-of-sale is data.
- Information embodies the understanding of a relationship of some sort between data. For example, analysis of point-of-sale transactions yield information on consumer buying behavior.
- Knowledge represents a pattern that connects information and generally provides a high level of predictability as to what is described or what will happen next. An example of knowledge is the prediction of promotional efforts on sales of particular items based on consumers' buying behavior.

Powerful *data-mining* engines were developed to support complex analyses and ad hoc queries on a data warehouse's database. Data mining looks for patterns among hundreds of seemingly unrelated fields in a large database, patterns that identify previously unknown trends. These trends play a critical role in strategic decision-making because they reveal areas for process improvement and business opportunity. Examples of data-mining engines are those from SPSS[1] (IBM) and Oracle.[2] These types of facilities are the foundation for online analytical processing (OLAP) systems.

Reporting Tools

Figure 3 shows a sample digital dashboard, a data reporting tool used by business managers to provide insight into their data using graphics and easily extractable information. The dashboard can provide insight into key performance indicators with colored lights, alerts, drill-downs, and gauges. The user sees a high-level view of a business process, which can be drilled-down for specific and detailed statistics.

Figure 3 – Sample Digital Dashboard

This level of detail is often buried deep in the enterprise's data, making it otherwise concealed to a business manager. For instance, with the digital dashboard shown in Figure 3, a knowledge worker clicking on an object will see the detailed information for that object, shown in Figure 4.

Today, many versions of digital dashboards are available from a variety of software vendors. These dashboards and other sophisticated reporting tools are the collective product of business intelligence systems.

Figure 4 – Sample Reporting Tool

[1]Please visit https://www-01.ibm.com/software/analytics/spss/ for more information on SPSS.
[2]Please visit https://www.oracle.com/solutions/business_intelligence/data-mining.html for more information on Oracle's data-mining engines.

Driven by information discovered by a data-mining engine, they give the business manager the information required to:

- **See** immediate key performance measures
- **Identify** and correct negative trends
- **Measure** efficiencies and inefficiencies
- **Generate** detailed reports showing new trends
- **Increase** revenues
- **Decrease** costs
- **Make** more informed decisions
- **Align** strategies and organizational goals

Data Marts

As corporate-wide data warehouses came into use, it became clear in many cases that a full-blown data warehouse was overkill for many applications. *Data marts* evolved to solve this problem and are a specialized version of a data warehouse. Whereas a data warehouse is a single organizational repository of enterprise-wide data across all subject areas, a data mart is a subject-oriented repository of data designed to answer specific questions for a specific set of users. A data mart holds just a subset of the data that a data warehouse holds.

A data mart includes all of the needs of a data warehouse – ETL, data mining, and reporting. Since a data mart deals with only a subset of data, it is much smaller and more cost-effective than a full-scale data warehouse. In addition, because its database is much smaller because it only needs to hold subject-oriented data rather than all of an enterprise's data, it is much more responsive to ad hoc queries and other analyses.

Data marts have become popular not only because they are less costly and more responsive but also because they are under the control of a department or division within the enterprise. Managers have their own local sources for information and knowledge rather than having to depend on a remote organization controlling the enterprise data warehouse.

Enterprise Application Integration

As online data warehouse technology matured, another solution for integrating enterprise-wide data appeared, called *enterprise application integration* (EAI). In many cases, the functioning of one enterprise system is significantly enhanced if it requests data or immediately receives newly generated data from other systems. For instance, an inventory-control system could be much more responsive if it has immediate access to point-of-sale data so that it could monitor product movement and remaining available quantities in real-time. A bank could authorize a loan to a customer while the customer is sitting with the loan officer by analyzing in real-time the customer's current business with the bank, his credit worthiness, and the current equity in his assets.

This capability requires that various legacy enterprise systems with different hardware, operating systems, and databases be enhanced so that they can talk to each other and exchange information in real-time or near real-time. The deployment of new business services, which were previously impossible, are enabled because the data and the applications that understood it are isolated from one another.

Intersystem Communication with EAI

EAI provides a means for linking such systems, in some cases without having to make sweeping changes to existing applications or data structures. In effect, whenever an event occurs in an application, it is sent via a publish/subscribe (or alternatively broadcast) mechanism to other applications in other systems that may be interested in this event. Likewise, one system requests data from another system in real-time via a request/response mechanism. There are two techniques for implementing EAI, adapters (Figure 5), and there are two techniques of data replication for implementing EAI (). These techniques all perform the transformation (the *T* in ETL) of data to a common format to be used by all systems.

Adapters

A*dapters* are specialized interfaces between diverse applications and allow those applications to directly communicate, in a form that is native to each application. Major application changes are typically not required,

assuming that the application or adapter vendor has already created/provided the translation and interface for that application. The adapters perform the necessary translation to a common communications protocol that is understood by the adapter at each end.

Figure 5 shows an example of application integration via adapters. A point-of-sale (POS) application communicates with an inventory application. Adapters on each system perform the necessary protocol translation between the applications native communication interfaces, enabling them to communicate without having to change programming.

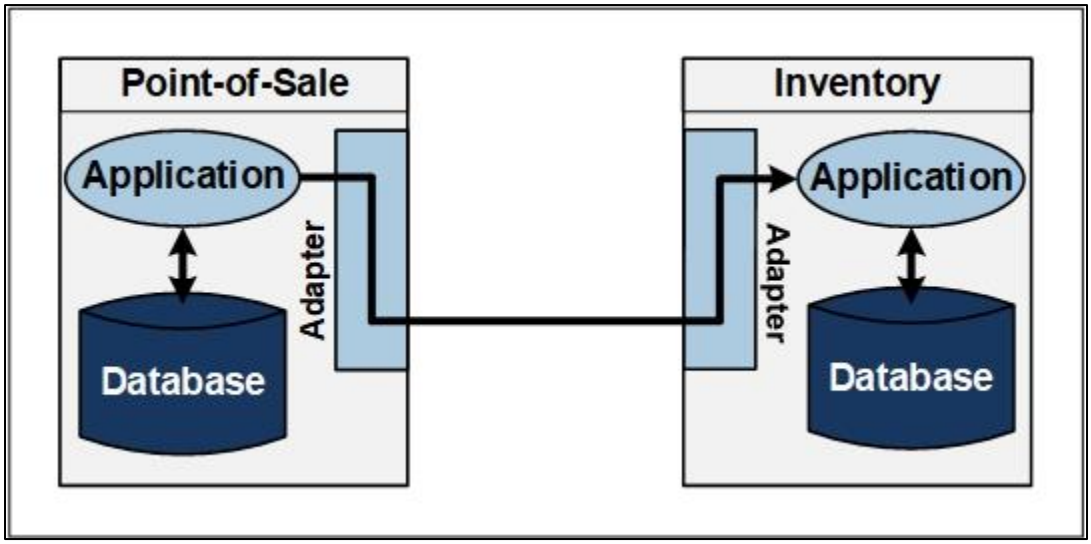


Figure 5 – EAI with Adapters

Adapter services must be defined by a common interchange data format to which all systems conform for their inter-application communication. As a consequence, the use of adapters is often invasive to applications, and can add inefficiencies to the message path length for the various translations that occur. Each application involved in communicating with other applications must be modified to use the common data format for the messages that it sends and receives, or an adapter must be developed to the application's existing interface. This problem is aggravated because knowledge is required of the proprietary formats used by the application vendors, information which many application vendors refuse to divulge. Consequently, adapter services can often only be implemented by the application vendor or in close cooperation with the application vendor.[3]

Also, each application may need to be modified to know how to select events to send, to which systems to send these events, how to process events it receives from remote applications, and how to respond to requests from remote applications.

Adapters are highly specialized because each application requires its own adapter that connects to the adapter network and that translates its application interface and data formats into the common formats of the adapter network. The range of applications that participates in an EAI network interconnected with adapters is often limited, and customized or "home grown" applications often do not provide the services necessary to interface. These problems all served to inhibit the widespread use of adapters for application integration, particularly for highly customized or "home grown" applications.

Message-Oriented Middleware (MOM)

Message-oriented middleware (MOM) provides a mechanism for asynchronously sending messages between applications. Messages are placed in queues and are sent to their destinations when the destinations are available. MOM is used to queue messages describing events and to send those messages to remote applications that take actions based on those events. MOM also provides the messaging facilities to send

[3]Of course, some applications either provide an adapter themselves or rely on one of the adapter vendors to create one or more for them. For example, TIBCO Software, Inc. provides adapter technology for integrating many of the popular application packages.

requests to other applications and to receive responses to those requests.[4] Figure 6 shows an example of application integration via MOM. The application examples used are the same as in Figure 5

In this case, the POS application puts messages into MOM queues. The MOM software then delivers the messages to other applications as configured (in this example, the inventory application).

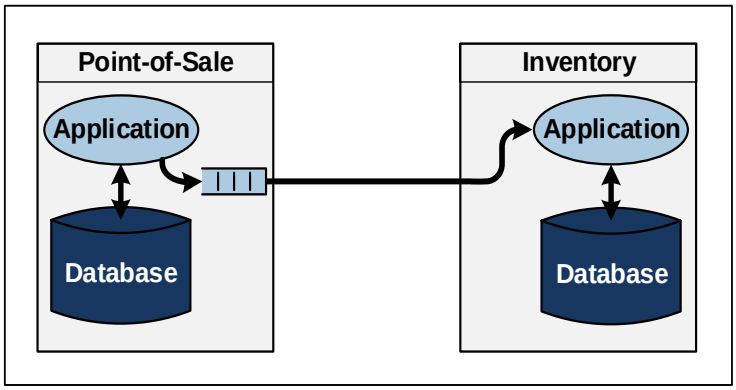


Figure 6 – EAI Intersystem Communication with MOM

However, the use of MOM in EAI suffers many of the same problems that adapters face. For one, MOM is invasive to the application; it can be architected and coded into new applications. However, retrofitting the MOM calls into an existing application requires access to the application source code, intimate knowledge of the application processing sequence, and extensive testing of the modified application. As with adapters, MOM requires that a common data format is established between all systems so that they can understand each other's communications. Applications must be modified to conform to this format when they communicate with other applications. In addition, applications must be modified to know when to send messages and how to respond to messages from other applications, as well as how to recover/continue processing when the MOM interface fails or becomes unavailable.

MOM interconnects typically provide persistence of the messages being sent and returned, allowing for their asynchronous nature. Of course, in some cases, these messages are sent synchronously, with the sending application often blocking or waiting for the reply. In such architectures, loss of the interconnect infrastructure, even temporarily, often means the application itself is down, or at least severely impacted. This issue can be an important design consideration, especially when the interconnect spans different systems or wide-area communications.

Data Replication

Data replication engines (in Figure 7 and Figure 8) exchange data between systems at the database level. This data may take the form of events generated by one system for consumption by other systems, or it may take requests from one system to other systems and the responses to those requests. Figure 7 shows an example of application interoperability via data replication, known as *data integration*. The same application examples are used as in Figure 5.

[4]MOM interconnects typically provide persistence of the messages being sent and returned, allowing for their asynchronous nature. In some architectures, these messages are sent synchronously, with the sending application often blocking or waiting for the reply. Loss of the interconnect infrastructure, even temporarily, often means the application itself is down or at least severely impacted. This issue can be an important design consideration especially when the interconnect spans different systems or wide-area communications.

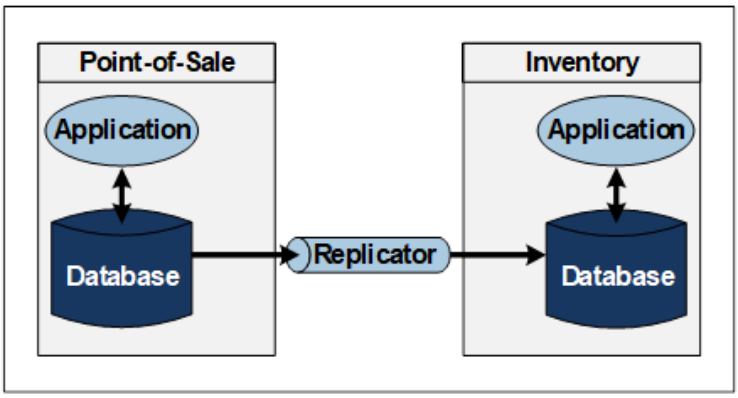


Figure 7 – EAI Intersystem Communication with Data Replication from Database to Database

In Figure 7, as the POS application updates its local database, the data replicator sends and applies these changes to the inventory database, where the data changes can be read by the inventory application and processed as appropriate. This approach is particularly useful for isolating target application processing from interconnection issues, since the receiving database is local to the target application environment. But, arguably, a more powerful approach is shown in Figure 8, where database changes in one database act as the real-time, event-driven feed into the other application. In order to interoperate with each other, applications may be integrated at the service layer or event layer.

Figure 8 shows an example of application interoperability via data replication, also known as *application integration*. The same application examples are used in Figure 5. This example also involves the use of a data replicator, which delivers the POS application changes directly to the inventory application, via an appropriate application interface, (instead of replicating the POS application data changes to the inventory database).
With *service-level integration*, special interfaces expose server-system data or application services to external client applications. One way to accomplish this is via specially designed agents that reside on the server system. For instance, a database agent may provide access to the target database via ODBC, JDBC, OLE, XML, ADO.NET, or Representation State Transfer (REST) technology.

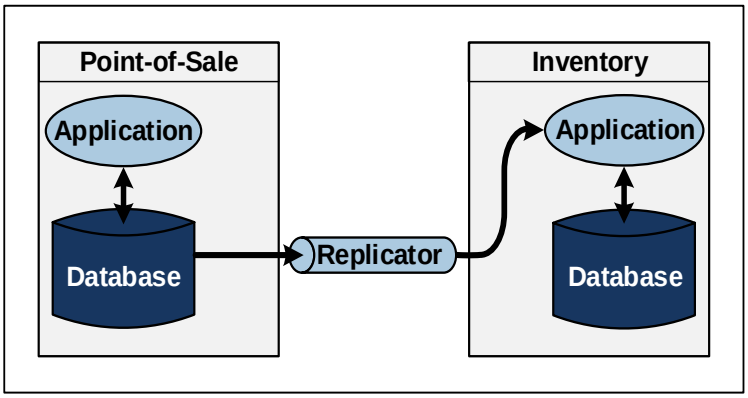


Figure 8 – EAI Intersystem Communication with Data Replication from Database to Application

Event-level integration provides event-driven integration services based on changes made to the source system's database. Event-level integration avoids the necessity of developing special agents or adapters for each application that is to be integrated. Rather, it monitors source application events in real-time and sends them immediately to the target application so that action can be taken on the event by the target.

Notice that this description has used the term "system" rather than "application," as was used in the descriptions for adapters and MOM. There is no integration with the data replication engine at the application level. Rather, replication works at the database level and occurs without any modification to the applications, which are typically unaware that replication is taking place. The only application modifications necessary are those required to implement new functions that make use of information from other systems (i.e., to take advantage of the availability of more information to more applications, enabling the rapid implementation of new business services).

A replication engine monitors changes to a system's database and is driven by a change log or by database triggers. Based on rules incorporated into the replication engine, certain database changes found in the change log are sent to other systems in the application network. Alternatively, database triggers can pass critical changes to the replication engine for dissemination to other systems. These changes are either placed in tables in the target systems (Figure 7) or are otherwise delivered to the applications via an interprocess message, MOM (Figure 6), or queue and may be then used by applications to implement new functionality for the enterprise.

In addition to the advantage of noninvasiveness described above, data replication has other benefits over directly using adapters and MOM. For one, with adapters or MOM, the unavailability of the network or another system brings down the application because it no longer has access to the data that it needs. With replication, applications continue to function in the presence of network failures or external system outages by using the local copy of the application database.

Data replication also brings locality to the application because the application has access to a local copy of the database. Performance is greatly enhanced, since it does not reach across the network for data. A common interchange data format is not required when using data replication engines, since the transferred data must be put into the target tables or target application's request format according to specified formatting rules. The transformation of source data to target data is performed by the data replication engine according to transformation rules specified by the user and incorporated directly into the engine. Applications access the new data or receive the new requests just as they would access any data or request.

Another advantage of data replication engines is that security is significantly simplified. With adapters and MOM, every application is an interface and must be secured via encryption. Data replication engines present only a single interface that must be secured. Not only is the management of security simplified, but the number of points that may be attacked are also reduced.

Sophisticated data replication technologies, such as the HPE Shadowbase product portfolio, also provide mechanisms to replicate data between systems, as well as feed database changes directly to applications by sending events to client applications or servers via queues or client APIs, which publish data to applications that have subscribed to the data,, or respond to poll queries from client applications. Since data replication avoids the application-specific problems associated with adapters and MOM, this mechanism has become the technique of choice for EAI.

EAI Networks

EAI networks may either be mesh networks or hub networks. In a mesh network (Figure 9), every system is potentially connected to every other system. It shows an EAI mesh network, with bi-directional arrows representing the flow of information among POS, inventory, gift registry, and sales promotion applications. Each application can publish, subscribe, request, and reply to any other application from data events or requests that are being transferred across the network.

A problem with a mesh network is the number of possible connections. If there are *n* systems in the EAI network, there can potentially be *n*(*n*-1)/2 connections if all systems have to communicate with all other systems. If a new system is added to the network, *n* new connections may need to be established, one to each of the existing systems. This number of connections creates a network complexity that rapidly becomes unmanageable as the network grows.

Figure 9 – EAI Mesh Network

A hub network eliminates this problem, as each system is connected only to the hub. Figure 10 shows an EAI hub network, with bi-directional arrows representing the flow of information from each application (POS, inventory, gift registry, and sales promotions) via the EAI hub, which routes each message to the appropriate target application.

Figure 10 – EAI Hub Network

A hub receives messages from each system and routes each message to the appropriate target systems; however, it is a single point of failure, and a hub failure disrupts all functions that require data from other systems. Thus, all benefits of EAI are lost during a hub failure. Since these messages are functions that are tactical in nature, governing the real-time response of the enterprise to events as they happen, enterprise operations may be severely compromised following a hub failure. Consequently, the hub must be made redundant so that it is highly available.

Another difference between mesh and hub networks is that in a mesh network, every system must know the data format of every other system. Each system must be able to transform its data format into those of the other systems. But In a hub network the hub is responsible for data transformation. Each system needs to only know the data format expected by the hub.

Nevertheless, a mesh network does have the advantage of avoiding a potential bottleneck at the hub for information exchange, particularly if the systems are located close to one another but the hub is remote. If message latency is of paramount importance, or if a hub design can be overwhelmed with request arrival rate spikes, then a mesh design may be appropriate for at least some of the connections.

EAI and Business Intelligence

Using these techniques, EAI allows applications to integrate on a real-time basis as events occur and as databases are updated without having to wait hours, days, or weeks for data from a data warehouse. Consequently, EAI supports tactical decision-making very well, but it does not provide the historical data needed for strategic decision-making.

Operational Business Intelligence

Operational business intelligence (OBI) systems provide an intermediate step toward satisfying the strategic needs that data warehouses address as well as the tactical decision-making that EAI addresses. An OBI system provides an event database that is frequently updated. As an historical database, the event log summarizes and satisfies strategic requirements. With frequent updates, strategic decision-making extends to daily or intra-day information that is used to immediately take operational action to address an immediate problem.

Industry experts in the field of business intelligence, noted that competitive pressures are forcing companies to react faster to changing business conditions and customer requirements. Business intelligence (BI) that helps drive and optimize business operations on a daily basis and sometimes used for intra-day decision-making, is called *operational business intelligence*. Since the objective of operational BI is to make more timely business decisions, it has a close relationship to *real-time* or *right-time* BI processing.

Indeed, OBI systems abound today. Conceptually, OBI systems are thought of as a data mart that is updated frequently (daily, every few hours, or even every few minutes or seconds) with mini-batches. OBI systems are similar to data marts because they generally focus on a specific task rather than on enterprise-wide functions. For instance, as shown in Figure 11, an OBI system might be periodically fed ATM and point-of-sale (POS) transactions. Its data-mining engine will then search for potentially fraudulent activity and prepare reports of such activity. After a review of these reports, a hold might be put on some credit or debit cards until the activity is resolved.

OBI systems bring together two needs for business intelligence, an historical database for strategic analysis and the capability to make rapid suggestions for operational actions. But what is still missing is the capability to make real-time suggestions for actions that are taken immediately upon the occurrence of some specified event; this capability is real-time business intelligence (RTBI).

Real-time business intelligence is also known as *event-driven business intelligence.* In order to react in real-time, a business intelligence system must react to events as they occur, and not minutes or hours later. After all, if one can use RTBI to stop suspected or fraudulent activity before it completes, enhance a shopper's online experience, and/or upsell additional products, RTBI has created real, measurable business value for the organization's bottom line.

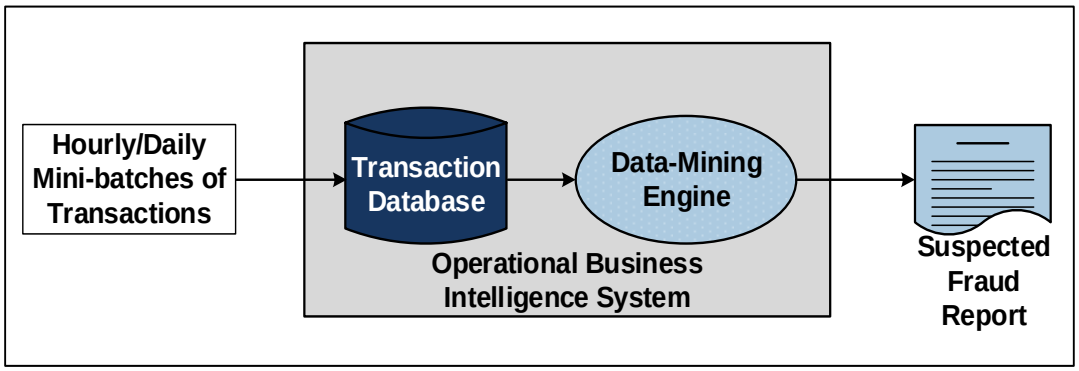


Figure 11 – A Fraud Detection OBI System

Real-Time Business Intelligence

The following sections discuss architectures that allow a business to react proactively in real-time to events as they occur rather than reactively at some time after they have happened.

Yesterday's Data is No Longer Sufficient

Data warehousing solves a strategic need of the enterprise because it manipulates massive amounts of data via data mining to derive new information and knowledge of an enterprise's operations. However, online data warehousing has little tactical value since the data in it is generally quite stale and can be days or weeks old. Data warehousing's primary value is in supporting strategic goals such as reducing costs, increasing sales, and improving profits. On the other hand, EAI solves a tactical need of the enterprise since it allows systems to react immediately to events generated by other systems; it has little strategic value since it provides no repository of data suitable for data mining.

A solution is needed that satisfies both the strategic and the tactical needs of an enterprise. A store, for instance, needs to know that men who purchase diapers on Saturday also tend to buy beer at the same time. A *strategic* way to capitalize on this information would be to put the beer near the diapers, and have beer sales on Saturday. A credit card company needs to know that a credit card being used to purchase an item in New York City was used thirty minutes earlier in Amsterdam. A *tactical* way to capitalize on this information would be to deny the transaction, and place a hold on the card.

Another example of the use of RTBI is in customer support. For instance, a customer often needs to know if he has enough credit or cash on his credit card or in his debit card account before attempting to make a purchase. Some card companies allow a customer to get preauthorization for specified amounts via cell phone to ensure that the purchase is covered, meaning that the card company must have up-to-date balances for its customers.

Today's Business Intelligence Needs

The gap between analytical and operational processing is closing fast. Just as Moore's Law[5] continues to characterize the rapid pace of technology development, the complex data-mining queries that used to take hours to run now execute in seconds. If only these data-mining engines had the latest values of the data, the tactical and strategic needs of business intelligence could be merged into a single solution. There are two primary impediments to effective and efficient RTBI: data latency and data unavailability.

Data latency refers to the *staleness* of data. The value of data degrades rapidly with its age. When people are relying on RTBI to tactically help them with on-the-spot decisions (real-time decision support), the freshest data and the fastest response times are needed.

[5] https://en.wikipedia.org/wiki/moore's_law

Data unavailability is a death knell for businesses. Business operations have progressed to the point that they are dependent on RTBI, and the unavailability of this intelligence due to a failed system could bring operations to a halt, so extreme availability of the RTBI services is paramount.
A business cannot respond to events as they happen if it cannot find out about these events for hours, days, or weeks (data latency). It also cannot immediately respond to events if the system that supplies the analyses of these events is down (data availability). If the problems of data latency and data availability are solved, businesses can react proactively to new information and knowledge rather than reactively. These problems are solved by sophisticated data replication engines such as HPE Shadowbase.

Real-Time Business Intelligence Systems

OBI systems represent a significant improvement in reducing data latency and enabling actions to be taken within hours of the events that triggered them; however they do not meet the criterion of immediacy that will allow a business to react in *real-time* to an event. For instance, an OBI system does not generate an offer to a customer while he is checking out at a cash register. Nor does it deny a potentially fraudulent transaction before it is executed.

An OBI system is needed that responds to events in seconds or less; but this response is not done by updating the OBI database with hourly mini-batches. Rather, the database must be updated with transaction activity in real-time as it occurs via change data capture (CDC); this type of update is called *trickle-feeding* the database (although a fast arrival rate may make it seem more like a fire hose is feeding the database). As transactions are received, they are stored and become a growing historical record of activity.

There must also be a very fast *rules engine* that analyzes incoming transactions against the historical database and makes decisions quickly enough so that immediate action of value to the enterprise is taken; this action is RTBI. There is a distinction between a data-mining engine and a rules engine. Both require an historical database; but a data-mining engine looks for relationships in the historical data to reactively support decision-making after the fact. A rules engine compares a real-time event to the historical data to proactively suggest an action to be taken.

The fraud-detection OBI system in Figure 11 can be extended into a real-time fraud detection system (Figure 12). In this figure, the RTBI system reacts fast enough to cause a suspicious transaction to be denied before it is consummated. Real-time ATM/point-of-sale (POS) transactions are fed to an RTBI system which then posts each transaction to its database for a rules engine to analyze. In real-time, the rules engine checks this transaction against recent activity on the credit or debit card and makes an instant determination of suspicious activity. It then generates a message indicating whether the transaction should be accepted or denied.

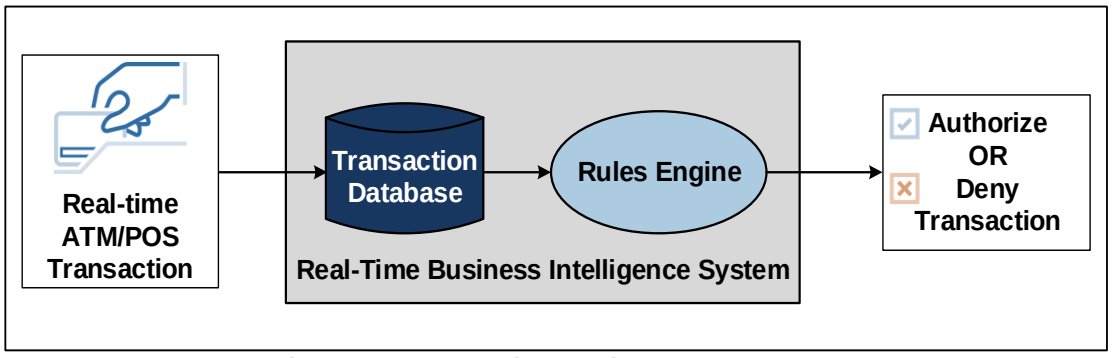


Figure 12 – A Fraud Detection RTBI System

Figure 13 shows an example of this particular fraud detection architecture. ATM and POS machines are generally serviced by a particular acquirer (bank), with the credit or debit card that was used typically being issued by a different bank. The card transaction must be sent to the issuing bank for authorization, and verification that the card balance is sufficient

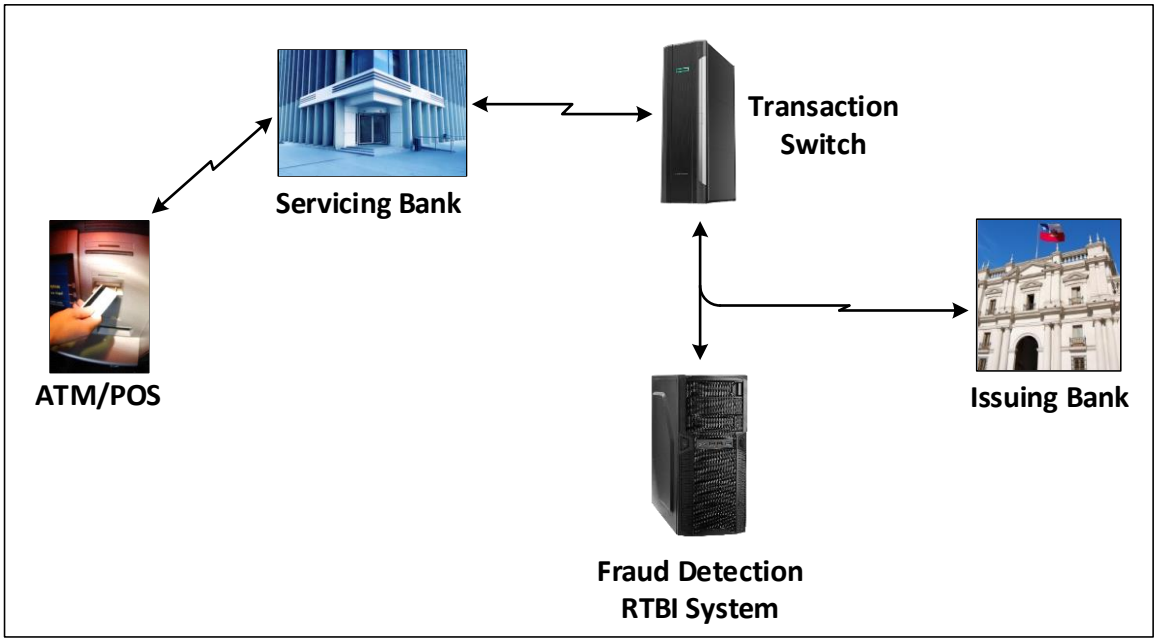


Figure 13 – A Real-Time Business Intelligence Example

In this case study, a bank service company operates a transaction switch that receives card transactions from the servicing banks operating the ATM or POS machines and forwards these transactions to the appropriate issuing banks. Upon receipt of an accept/deny message from the issuing bank, the transaction switch returns this message to the servicing bank, which then takes appropriate action to accept or deny the transaction.

Additionally, the bank service company provides a fraud detection service. At the same time that it forwards the transaction to the issuing bank for authorization, it also sends the transaction to its RTBI fraud detection system. If this system determines that the transaction is suspicious, the bank service company takes several actions as requested by the issuing bank. It certainly denies the transaction and informs the issuing bank, but it may also alert the issuing bank to the circumstances of the suspicious activity so that it can decide whether to put a hold on the card. This system shows RTBI in action as the RTBI fraud detection system uses complex rules against an historical database to determine which actions to take that directly affect a transaction in progress.

RTBI Dashboards

RTBI dashboards are used to bridge the gap between operational business intelligence and real-time business intelligence. For instance, Figure 14 shows an IT server network-monitoring dashboard, which displays not only historical information but also shows the current status of the server network. The dashboard is interesting because it performs all three business intelligence functions, strategic, operational, and tactical.

From a strategic viewpoint, it shows bandwidth usage and connection quality over the last several weeks, and by clicking on the History tab, further historical usage and quality data are displayed. This dashboard forms the basis for future planning of network upgrades. From an operational viewpoint, the dashboard shows the current status of the network, the current memory usage, and the current connections and their traffic. If a network problem occurs, operational staff could immediately take remedial action. From a tactical viewpoint, as problems are detected, the RTBI system driving the dashboard can issue audible, email, or cell phone alerts to the operations staff. The RTBI system even takes automatic remedial action such as rerouting network connections, shedding low-priority traffic, or invoking redundant connections. The transactions are fed to the new RTBI system from the enterprise's other systems with the *online ETL*.

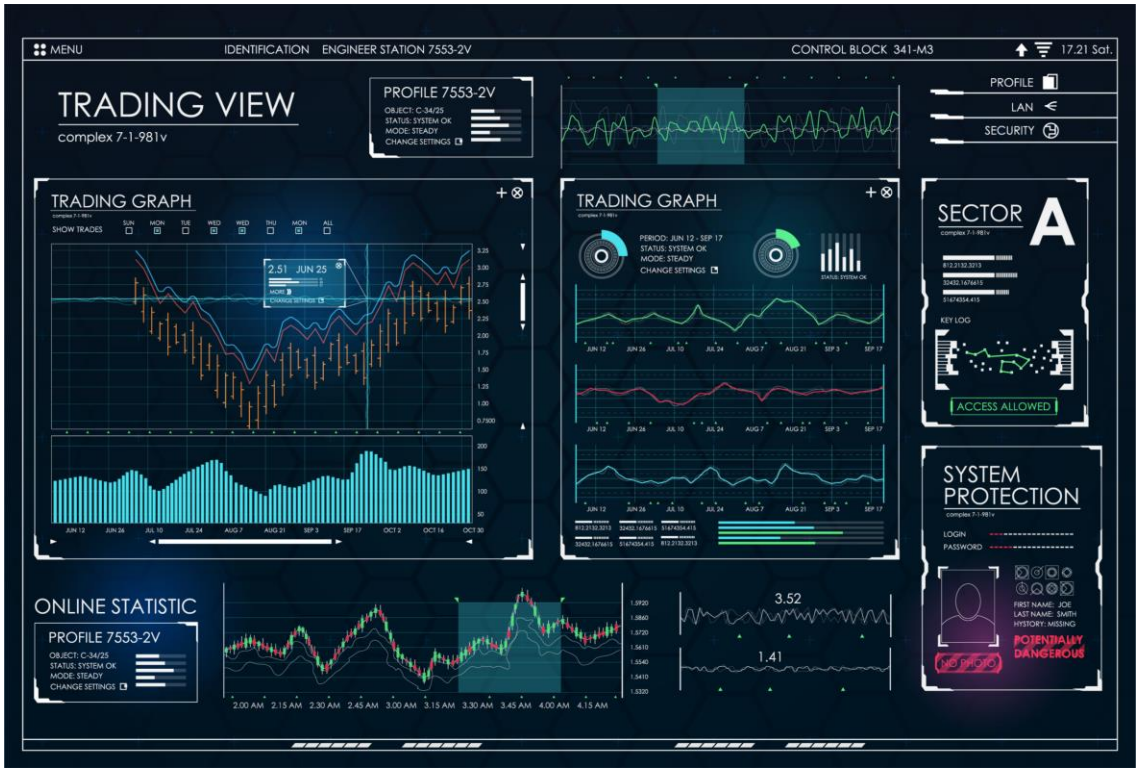


Figure 14 – Sample RTBI Dashboard

Online ETL

ETL is the facility that allows data to be *extracted* from a source database, *transformed* into a common format, and *loaded* into a target database (the data warehouse's database). Since contemporary ETL facilities are batch-oriented and run periodically, they are characterized as being *offline* ETL facilities. EAI exchanges current information between systems in an application network but provides no historical record of enterprise activity for strategic-analysis purposes.

RTBI needs an *online* ETL facility that not only preserves historical strategic data but can also provide current tactical data. The online ETL's job is to create and maintain a synchronized copy of a source database on a target database (the RTBI system) *while* the source database and the target database are actively updated and used by multiple applications. In effect, as transactions occur in the enterprise, they are trickle-fed to the RTBI system in such a way that this activity is transparent to other ongoing operations. As with EAI, three methods can be used to create an online ETL facility – connecting via adapters, using message-oriented middleware, and synchronizing via low-latency replication engines.

Adapters

Early adaptations of RTBI used an extension of existing EAI technology. Adapters were used to interconnect enterprise systems with the RTBI system. As transactions were executed by an external system, the results of those transactions were communicated to the RTBI system via adapter connections. The adapters also serviced requests from external systems and returned the RTBI system replies to those systems.

However, adapter technology suffered from the same problems that it faced in EAI applications. It was invasive and often required application modification in order for the applications to interface with the adapters. In addition, adapters were specialized to the applications. Each adapter knew the proprietary formats of the application data structures and how to interface to its application and was thus custom designed for that application. Every time the application changed, the adapter was modified, so consequently, not all applications could participate in the online ETL function.

Message-Oriented Middleware

MOM also exhibited many of the same problems that plagued adapters. MOM was invasive and required changes to the application to send and receive appropriate messages in a common interconnect data format, or to use adapters. In order to make application changes, access to the application's source code was required,

but this source code was often lost or was the proprietary property of a third-party vendor, and so was not available. Additionally, every time the application changed, the MOM code potentially had to be modified. As with the EAI implementations, adapters and MOM RTBI implementations also suffered from the problems related to network interconnect issues. Due to the immediate nature of RTBI, if the interconnect went down, the RTBI response was either delayed, or worse, failed altogether.

Data Replication

Data replication is synchronous (data is replicated to the target system concurrent with the application data changes) or asynchronous (data is replicated to the target system some very short time after the application has made its changes). In RTBI applications, asynchronous replication is generally used, allowing the replication activity to be totally transparent to the application. The application proceeds with no knowledge of or impact from the replication activity and the application's database activity is extracted by the replication engine via a transaction log, triggers, or intercepts. Selected updates are sent to the RTBI target system by the replication engine, where they are applied. The replication latency interval, which is the time that it takes to propagate a source database change to the target database, is measured generally in sub-seconds.

Data replication solves the adapter and MOM problems of application invasiveness and specialization, as well as data availability, in an RTBI environment. Because a data replication engine (Figure 15) is non-invasive and is application-unaware, it only deals with the database, and is isolated from the application by the database. Today's replication engines support most relational databases and many non-relational databases as well. An additional benefit is that the same data replication engine, such as HPE Shadowbase, can also serve other purposes at the same time as fulfilling the RTBI mission, for example, providing business continuity and disaster recovery.

Figure 15 depicts data consolidation. An application makes changes to a source database, which are captured in a transaction log and then triggers the replication engine to transform and send the change data capture (CDC) event information into an RTBI database.

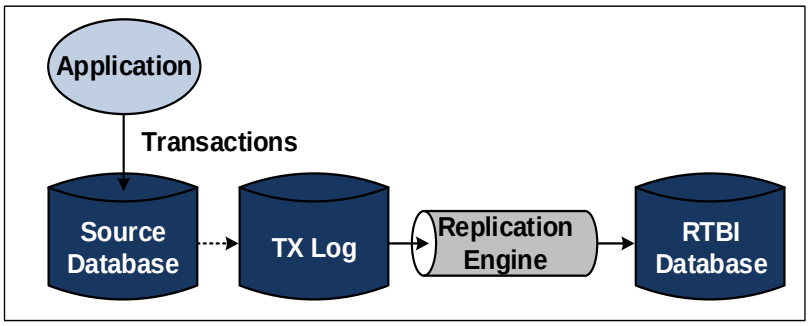


Figure 15 – Data Consolidation via Data Replication

Data replication is synchronous (data is replicated to the target system concurrent with the application data changes) or asynchronous (data is replicated to the target system some very short time after the application has made its changes). In RTBI applications, asynchronous replication is generally used, allowing the replication activity to be totally transparent to the application. The application proceeds with no knowledge of or impact from the replication activity and the application's database activity is extracted by the replication engine via a transaction log, triggers, or intercepts. Selected updates are sent to the RTBI target system by the replication engine, where they are applied. The replication latency interval, which is the time that it takes to propagate a source database change to the target database, is measured generally in sub-seconds.

Replication engines support rules for data transformation. Since some rules are built in, and others are specified by user-supplied routines, data transformations of any kind are possible without modifying the applications, the databases, or the core replication engine.

Since replication engines are required to preserve transactional consistency, source updates are not randomly applied to the target database. Transactions must be applied in the same order to the target database that they were made at the source database.

An additional problem, called a *data collision*, occurs when multiple sources simultaneously update their copy of the same data item. For instance, what happens if two enterprise systems update the same customer address with different data at the same time? Which one is correct?

Data collisions may happen with any asynchronous mechanism used to feed the RTBI system, whether it is replication, adapters, or some other form of messaging. Data collisions do not occur if synchronous replication is used. Some replication engines are particularly adept at handling collisions, detecting and resolving a collision via specified rules. Some of these rules are built in, and more application-dependent rules are added via user-supplied routines. HPE Shadowbase is able to determine where a data collision will happen before it occurs and has data collision handling settings that allow the user to specify how to handle data collisions.

Another problem can occur if a target system fails, and during its downtime, other external applications try to send it data. If an application does not get a response to its sent data, the application may fail or its data may be lost. However, transactional replication engines queue this data and will send the data updates to the system when it is returned to service. As discussed below, in the unlikely event of an RTBI system failure, the replication engines in each enterprise system queue their data changes and send them to the RTBI system when it is returned to service, and no valuable business data is lost.

A similar problem can occur when a communication connection is lost. With data replication, a system that loses its connection with an RTBI system continues to operate in so-called “split-brain mode.” It is unaffected except that it does not send its data updates to the RTBI system, and it does not receive updates or recommended actions from the RTBI system. When the connection is restored, all of the updates that accumulated in either direction are sent to the opposite system via the queues maintained by the replication engines. In split-brain mode, each system applies updates to its local copy of the data invisibly to the other system while the connection is down; so there are bound to be data collisions. These collisions are resolved on-the-fly by the replication engines after recovery of the network and during the resynchronization of the databases.

Online Copying

Another requirement for online ETL is an online copy utility, which is needed to bring an RTBI system into operation. An up-to-date snapshot of the data in the various enterprise systems that will feed the RTBI system must first be loaded into the RTBI system before it becomes effective. This load must occur without affecting the source systems since they are busy running the enterprise and must include all the various changes that occur during the copy, which could take hours or even days to complete.

Additionally, the copy must include all of the transformations that were otherwise made by the online ETL facility in order that the initial RTBI database properly reflects the state of the enterprise. The HPE Shadowbase SOLV product (built by Gravic, sold by HPE) meets all these requirements.

Extreme Availability

Once an enterprise activates RTBI, it would suffer greatly if it lost this capability. Instant reactions that made it competitive and efficient would suddenly be lost. Extreme availability of the RTBI system is of paramount importance.

The first step is to choose an architecture that is especially resilient to failure. NonStop systems from HPE are an ideal solution. A single NonStop system has proven availabilities in excess of four 9s (99.99% availability, equating to approximately 52 minutes and 32 seconds of downtime per year). Adding geographic redundancy with HPE Shadowbase typically increases the availability to eight 9s. Clusters of highly reliable industry-standard servers using redundant arrays of independent disks (RAID) or mirroring storage are another choice and are configured to provide availabilities slightly less than HPE NonStop systems, but are much more complex to manage and generally have a higher *total cost of ownership* (TCO)[6].

Though these solutions give reasonable protection against single component failures, they do nothing for disasters that take out an entire data center. The RTBI system must be backed up by a geographically remote site that takes over in the event of a primary site failure; this is called a *fault tolerant* solution. Otherwise, it

[6]For more information on TCO, please reference *Why Your Business Continuity Plan May be Inadequate*.

might take days, weeks, or even longer to replace the system, during which time normal business operations are severely impacted.

Backup systems using magnetic tape to rebuild the database take days to recover and are unsuitable for RTBI system backup. Virtual tape eliminates some of the problems associated with magnetic tape, but virtual tape systems still take a prohibitively long time to recover. Data replication to a backup site provides a reasonably complete copy of the database, but following a primary site failure, the database must still be brought to a state of consistency, the applications started, the database opened, and the system tested before it is returned to operation. This strategy takes some time (minutes to hours) and is hindered by the same problem that the other backup strategies face, which poses the question: will the backup system properly come up when it is brought into operation?

The best "backup" for an RTBI system is a continuously available, active/active system, which provides for continuous availability.[7] An active/active system comprises two or more geographically-dispersed nodes that are already up and running, with each node actively processing and sharing the application load with the other nodes. If a node fails, transactions (or users) must be switched from the failed node to the surviving nodes, a switch that can be done in seconds.

The primary advantages of having an RTBI active backup are twofold. First, failover is accomplished in seconds. Users of the RTBI facilities may not even know that a failure has occurred. Second, the failover process itself does not fail. Since the backup system is already up and running, it is known that it is fully operational, and its operation is being verified with every transaction it processes.

For some applications or other operational reasons an active/active RTBI configuration may not be possible. An alternative configuration that may be used in such cases is known as *sizzling-hot-takeover* (SZT), also known as *sizzling-hot-standby*. The only difference between this alternative and an active/active configuration is that the standby system is not actually processing transactions, in all other respects it is identical, and shares most of the advantages of an active/active system but without the issue of data collisions.[8]

As previously stated, there are two impediments to RTBI, data latency and data unavailability. Online ETL (and especially data replication) solves the problem of data latency. Active/active or SZT RTBI systems solve the problem of data unavailability.

The Operational Data Store – The Next Evolutionary Step?

What is an Operational Data Store?

In the early 2000s, the Gartner Group coined the term zero latency enterprise (ZLE). In its words, ZLE was "the instantaneous awareness and appropriate response to events across an entire enterprise." This response was later renamed the real-time enterprise (RTE). HPE was a leader in ZLE, with its ZLE architecture centered around an operational data store (ODS) that was similar to the data store used by an online data warehouse. Rather than periodically reloading the ODS with massive amounts of data via an ETL facility, the ODS was trickle-fed transactions as they occurred so that it always contained the latest state of the enterprise as well as historical information (Figure 16). Using the ODS, classical data-mining engines could generate strategic information and knowledge, and real-time rules engines could make tactical decisions regarding immediate actions to take.

[7] For more information on active/active systems, see the Gravic white paper, *Achieving Century Uptimes with HPE Shadowbase Active/Active Technology*.

[8] Active/active configurations have additional benefits such as better utilization of all systems, but SZT is a much better solution than an active/passive architecture.

Figure 16 shows an operational data store (ODS) network, with bi-directional arrows representing the flow (replication) of information between the ODS and point-of-sale (POS), gift registry, inventory, strategic activities for a decision support system (which includes ad-hoc queries, reports, and dashboards), sales promotions, and rules engine (which sends the data to tactical, rules-based applications). In this architecture, any application can publish, subscribe, request, and reply to the ODS, which consolidates, extracts, and transfers rich, up-to-date information from any other part of the network.

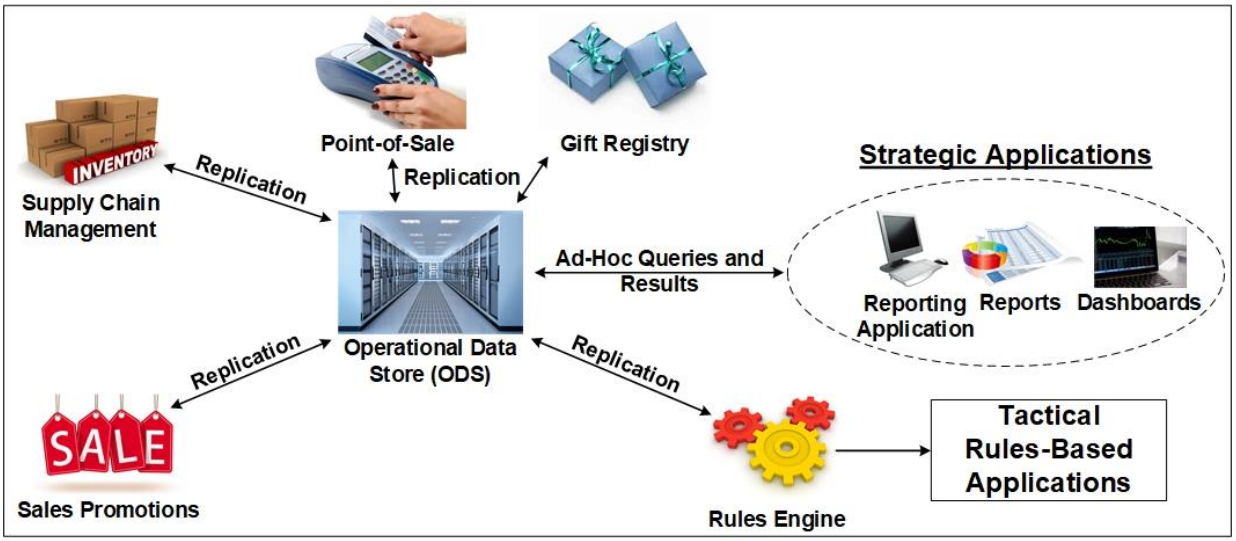


Figure 16 – The Operational Data Store

One particular ODS characteristic requires it to handle mixed workloads. On the one hand, it must be able to respond to complex queries from knowledge users, data-mining facilities, and rules engines using *online analytical processing* (OLAP). The database structures suitable for OLAP are characterized by *fat keys* that allow rapid searching of the database to respond to complex queries. On the other hand, the ODS must be capable of *online transaction processing* (OLTP) at an extremely high transaction rate, as it is being fed transactions in real-time from many enterprise systems. The database structures suitable for OLTP are characterized by *skinny keys* that require a minimum of updating as data is added to the database.

Another particular ODS characteristic is that it is bi-directional. Unlike a data warehouse, which typically only accepts information from enterprise systems, an ODS both accepts information from and delivers information to the other enterprise systems. An example of this characteristic is the act of keeping databases in synchronization. A particular data item, like a customer's address, may be stored in several databases around the enterprise. If one system changes this data item, the ODS acts as a central data repository that informs the other systems of the new data value so they update their databases.

Other examples of outgoing information are the results of the rules engine. If the rules engine decides to recommend a particular immediate action, that action is communicated to the appropriate enterprise system for execution. For instance, if the rules engine for a credit card processor detects suspicious activity, it immediately alerts the authorization system to take appropriate action.

In concept, the ODS, which contains all of a corporation's data, could become the *database of record*, the single version of truth. This action generally does not happen because of regulatory requirements and other considerations and the database of record remains on the existing systems, where it was resident for decades.

The Evolution of RTBI to ODS

As previously described, the ODS is simply the RTBI system but extended to the enterprise. Though RTBI systems are becoming common today, full-blown ODS systems have yet to make a significant appearance due to the complexity and cost of designing and building such far-reaching systems.

Figure 17 shows the evolution of the ODS, from its early stages (top of the diagram) to data consolidation (present day, bottom of the diagram). In its early stages, an ODS consisted of separate tactical and strategic systems, populated via periodic batch updates. These systems evolved into real-time systems, with ODS data being updated as it was changed on source systems, but there were still separate tactical and strategic systems

with multiple ODS databases. There was no single, consolidated view of the enterprise. At the bottom of the figure, data from all these multiple ODS sources is consolidated in real-time into a single, true ODS, driving all the tactical and strategic decision making across the enterprise.

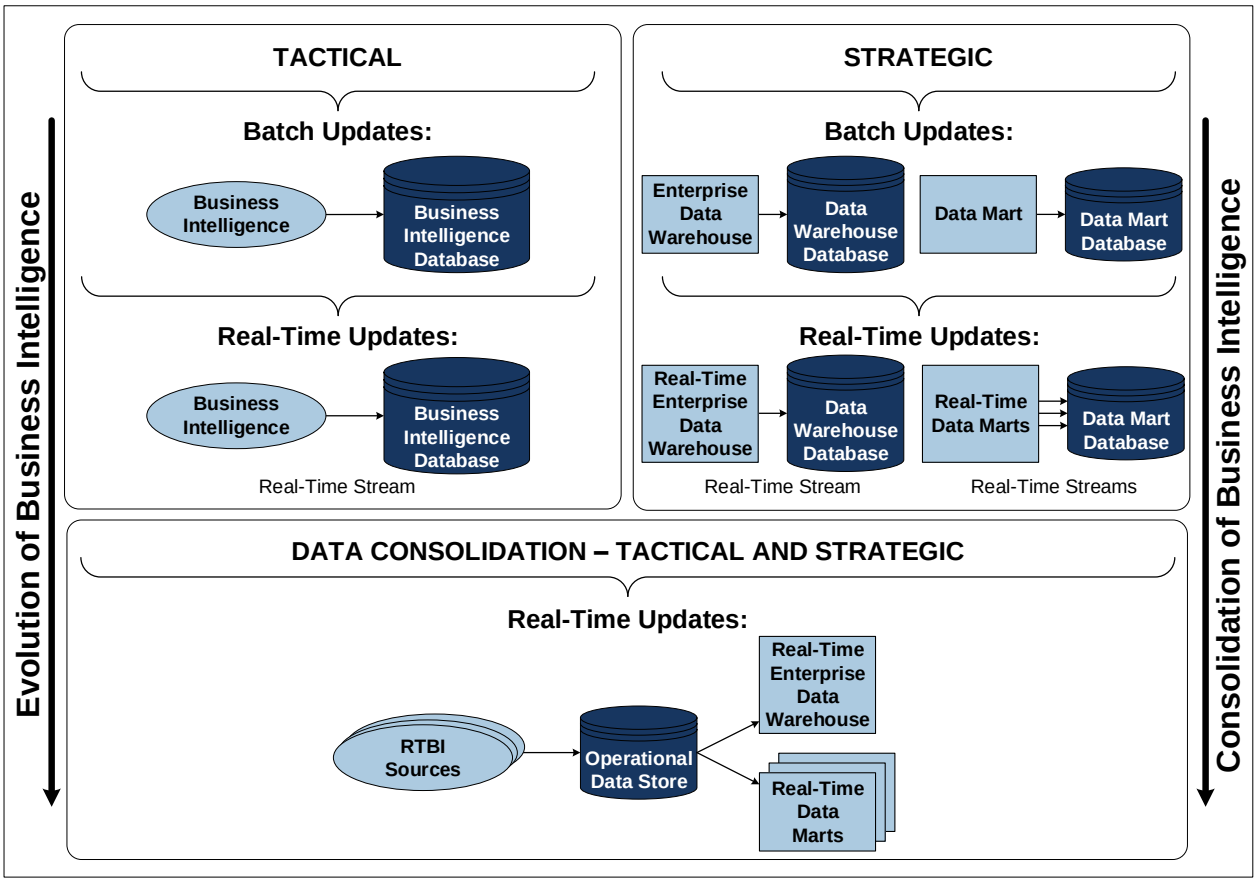


Figure 17 – The Evolution of the Operational Data Store

It is conceivable that RTBI systems may evolve eventually to ODS systems, as shown in Figure 17. Early dedicated business intelligence systems used data warehouse or EAI technologies. As these technologies proved their worth, data replication was added to move the technologies to RTBI systems, thus expanding their reach. Though RTBI systems exist today in many applications, each RTBI system generally supports a single purpose such as fraud detection, instant customer promotions, or just-in-time inventory.

Considerable effort was invested by some companies to consolidate a multitude of RTBI systems into a single ODS supporting enterprise-wide tactical and strategic decision-making or enterprise content management, shown as the final step in Figure 17.

The advantages of such integration are clear:

- **The single ODS supports both tactical and strategic** decision-making.
- **The ODS is made highly available** through redundancy, such as using an active/active system to achieve not only high availability of the system, but also to achieve disaster tolerance. Corporate functions are less affected by the failure of one of the other systems. For example, a customer's credit status is checked against the ODS without having to interrogate a credit-authorization system that might be down.
- **The scope of decision-making is extended** to many more areas across the enterprise. For instance, a drop in sales is correlated with increased accounts receivables on store-branded credit cards. Easier credit terms might help to restore sales to their previous level.
- **The various corporate IT systems are isolated.** No longer do they have to interact with each other through EAI. They each communicate only with the ODS system. Newly added applications do not have to be configured to interface with multiple other applications and only need to interface with the ODS. Also, other systems do not have to be modified to interface with the new system.

However, so far, the obstacles to achieving this goal have been daunting. For instance:

- **The ODS is an expensive system** in terms of hardware acquisition, software licensing, and development.
- **More powerful data-mining engines** and rules engines may need to be incorporated.
- **The design of the ODS database is much different** and needs to support both tactical and strategic queries, yet be very efficient in handling a large volume of updates. Fast update processing requires skinny keys in which a minimum of alternate indices must be updated, while efficient query processing requires fat keys providing many access paths to the data.
- The size and depth of the ODS can be quite extensive, as it typically has to store large volumes of historical and archival versions of the data, instead of just the current value of the data (also referred to as managing *long data* issues).
- Current database products tend to handle structured data well, but corporations also want to leverage their unstructured data (e.g., video, email, voice, text) for business gain. So-called *big data* collection, indexing, and accessing issues tend to thwart co-locating all of this information into a single repository, managed by a single DBMS. The volume of data generation overwhelms typical storage capacities. Clearly, more powerful data collection, transmission, storage and retrieval systems are needed.
- Applications may have to be significantly rearchitected, which is not only expensive and time consuming, but risky.
- Third-party products that do not readily lend themselves to adapting to an ODS architecture may be involved.
- The conversion of current decision-making processes might not only be difficult but may, in fact, be resisted by the user community.

The bottom line is that today, companies are achieving RTBI by directly integrating their systems using real-time heterogeneous data replication, or they are trickle-feeding data marts in real-time and using these marts to gather information. These warehouses or application networks may or may not turn into an ODS as consolidation occurs. If no warehouse currently exists to act as the stepping stone to an ODS, companies may find it more economical to simply interconnect their systems in a mesh network using existing EAI technologies, instead of following a more planned and fruitful, but expensive path to an ODS.

HPE Shadowbase Real-Time Business Intelligence Solutions

This paper has laid out the evolution of RTBI, and the various technologies that have been used over the years to achieve it. Of these technologies, data replication is by far the most flexible, least disruptive, and easiest to implement. The HPE Shadowbase Product Suite provides the replication and online copying facilities needed for the implementation of effective RTBI systems. These products also provide the infrastructure necessary to implement continuous availability architectures including the HPE Shadowbase real-time data replication engine and the HPE Shadowbase Online Loading (SOLV) copy utility.

HPE Shadowbase Software – Real-Time Replication

The Shadowbase data replication engine provides high-speed data replication among a variety of databases and platforms. With a history in the decades and hundreds of large, high-volume, mission-critical, enterprise-class installations worldwide, the Shadowbase engine is a proven performer and is an ideal solution for interconnecting enterprise systems with an RTBI system.

HPE Shadowbase Software

- **Provides bi-directional asynchronous data replication**, supporting active/passive, Sizzling-Hot-Takeover (SZT, also known as Sizzling-Hot-Standby), and active/active system configurations. Shadowbase synchronous data replication is also available[9].
- **Is totally transparent to the applications**. It requires no application modifications because it gets database changes from a transaction log or from database triggers or intercept libraries (real-time change data capture).

[9]For more information on active/active systems and determining which architecture is best for you, please see the Gravic white paper, *Choosing a Business Continuity Solution to Match Your Business Availability Requirements*

- **Runs on a wide variety of platforms against several databases;** source platforms include HPE NonStop, Linux, Unix, and Windows. Target platforms include all source platforms plus OpenVMS and AS400. Shadowbase software replicates from NonStop SQL, Enscribe, Oracle, Sybase, and SQL Server databases to these same databases as well as to DB2 and MySQL databases, and can also replicate to any ODBC-compliant target database. (Please see our current list of supported databases and platforms.[10])
- **Has no disk-queuing points, thus, replication is very rapid.** Replication latency is measured in the tens of milliseconds on many systems and performance is enhanced by its multithreading capability to handle massive loads.
- **Runs as check pointed process pairs and persistent processes** on HPE NonStop systems so that it survives hardware and software failures. On other systems, its processes run as persistent processes under a monitor.
- **Guarantees the integrity and consistency of the target database even when running with multiple threads.** It is configured to break source transactions into sub-tractions to enhance replay parallelism and performance.
- **Has extensive built-in data transformation rules** that are augmented by user-supplied specialized rules.
- **Provides complete data collision detection and resolution facilities for active/active environments.** Users extend its collision resolution rules with additional business rules tailored to their application's business logic.
- **Is heterogeneous;** any supported platform and any supported database may serve as a source system or as a target system in any mix.
- **Is easy to manage with its AUDMON process monitor,** its AUDCOM command interface, and its Shadowbase Enterprise Manager (SEM), a Windows GUI that provides integrated command and monitoring support for its components running in a heterogeneous environment.

Extreme Availability with HPE Shadowbase Active/Active Support

Many HPE Shadowbase installations around the world are active/active, geographically distributed systems. These systems achieve availabilities in excess of six 9s (less than thirty seconds per year of downtime). Shadowbase replication provides two functions in an RTBI environment: it synchronizes the operational systems with the RTBI system and ensures the availability of the RTBI systems, via an active/active or Sizzling-Hot-Takeover configuration.

Running an RTBI system as an active/active system simply requires that the Shadowbase engine replicates database changes between the two geographically separated RTBI systems. If one system fails, the other system is immediately ready to take over the RTBI services. With an active/active RTBI system, it is fair to say that the effective uptime of RTBI services is measured in centuries.

HPE Shadowbase SOLV – Online Copying

The critical online copying function needed to bring an RTBI system into service is provided by the unique copying utility, HPE Shadowbase Online Loading (SOLV), which:

- **Moves data from a source database to the target database** without interrupting source database activity
- **Maintains target data that was already copied in a current state** by applying changes to it as other portions of the source database are copied. There is no large queue of source database changes that needs to be applied following the copy
- **Maintains currency of those parts of the database** that were already copied, which may then be used by applications at the target system while the copy is in progress

Another powerful feature of SOLV is inherent in its architecture. Since SOLV is not a utility separate from the HPE Shadowbase replication engine, it uses the Shadowbase replicator to move data from the source database to the target database. As a consequence, SOLV uses the same data transformation rules during the copy function that the Shadowbase engine uses during later data replication. This function avoids the problem of trying to conform the copy transformation rules with the replicator transformation rules so that both

[10]For more information on active/active systems, see the Gravic white papers, *HPE Shadowbase Total Replication Solutions for HPE NonStop* and *HPE Shadowbase Total Replication Solutions for Other Servers*.

perform exactly the same. Rules are implemented only once because the same code is then used by both the load engine and the replication engine.

HPE Shadowbase Streams – Data and Application Integration

Shadowbase Streams, another member of the HPE Shadowbase product line, streams data generated by one application to other applications and facilities as well as target database environments. Shadowbase Streams for Data Integration and Application Integration provides the facilities for integrating existing applications at the data or event-driven level in order to create new and powerful functionality for the enterprise.

HPE Shadowbase Streams:

- **Quickly and easily integrates changes** made in any database into other data environments to keep that target information synchronized. The changes can be made in real-time or batched for periodic snap-shot or micro-batch updating
- **Enables interoperation between applications** that were once isolated in an event-driven fashion in real-time. Critical data generated by one application is distributed and acted upon immediately by other applications
- **Supports many models for data distribution,** including maintaining a remote database of critical data, sending critical data to client applications or servers directly via queues or client APIs, publishing data to applications that have subscribed to the data, and responding to poll queries from client applications.
- **Is extensible, allowing the user to embed custom processing logic** into the data processing path. It readily filters, transforms, and adapts data from one application or database environment into the format or protocol required by another application or database environment, all without requiring any changes to the existing application that is generating the data
- **Modernizes legacy applications by integrating diverse applications** across the enterprise so that new and valuable services are generated to enhance competitiveness, to reduce costs or to increase revenue, to satisfy regulatory requirements, and to generally improve the user experience
- **Plays a significant role in the distribution of data** from disparate sources throughout the enterprise in real-time, thereby facilitating the implementation of RTBI capabilities[11]

HPE Shadowbase Real-Time Business Intelligence Case Studies

HPE Shadowbase solutions are used extensively to provide RTBI to a variety of enterprises. Some examples of these applications and how it is being used to improve business services, company competitiveness, and profitability are shown in the case studies below.

[11]For more information, please see the Gravic white papers, *HPE Shadowbase Streams for Data Integration* and *HPE Shadowbase Streams for Application Integration*.

Fraud Detection

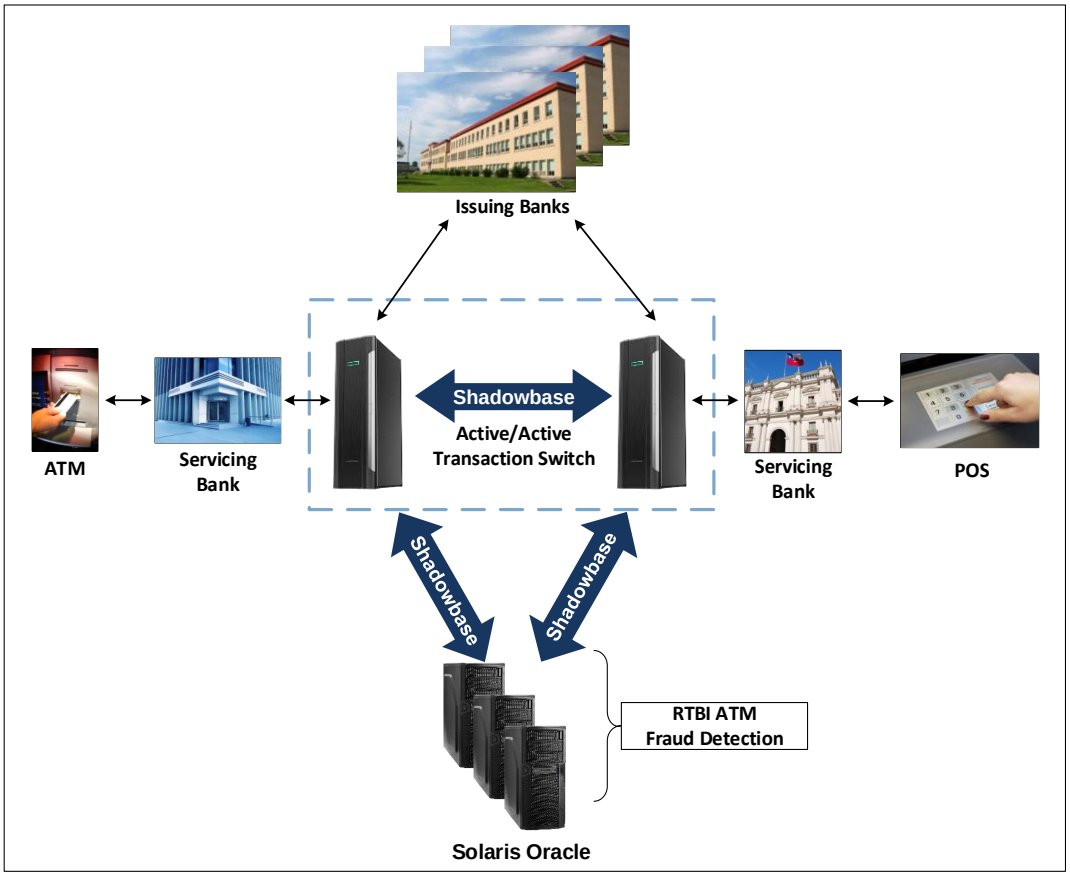


Figure 18 – Real-Time Business Intelligence for Fraud Detection

ATM and point-of-sale (POS) devices are typically managed by a local servicing bank. However, this bank is likely not the bank that issued the credit or debit card used at the device. A major provider of interbank electronic transaction switching services for ATM and POS terminals routes each credit card transaction from the servicing bank to the bank that issued the credit card. The issuing bank then returns an authorization or a rejection to the credit card terminal via the provider's switching system and the servicing bank.

If the transaction switch fails, no credit card transactions are accepted by the terminals that the transaction switch provider services. The provider uses an active/active, geographically distributed system comprising HPE NonStop nodes to provide extreme reliability for its transaction message switch (Figure 18). Shadowbase bi-directional replication is used to keep the databases of the active/active nodes in synchronization.

As an additional service to the credit card companies that are its customers, the provider offers fraud detection. Credit card transactions captured by its NonStop servers use the Shadowbase data replication engine on a Solaris server running Oracle. The credit card transactions are transformed by Shadowbase replication to the format required by the Oracle database. Complex queries are run against the Oracle database to monitor card activity for suspected fraudulent activities.

If a suspected fraudulent transaction is detected, an indication of this transaction is immediately returned via Shadowbase replication to the NonStop switch for forwarding to the issuing bank. The issuing bank can elect to reject the transaction and deactivate the card or to allow the transaction.

This case study is an excellent example of RTBI at work. The Solaris system is an RTBI system and maintains a history of credit card usage for each credit card and uses the current credit card transaction data to deduce the possibility of fraud. If fraud is suspected, this suspicion is communicated in real-time to the issuing bank for immediate action, potentially saving the company thousands of dollars in fraudulent transactions.

Investment Product Promotion

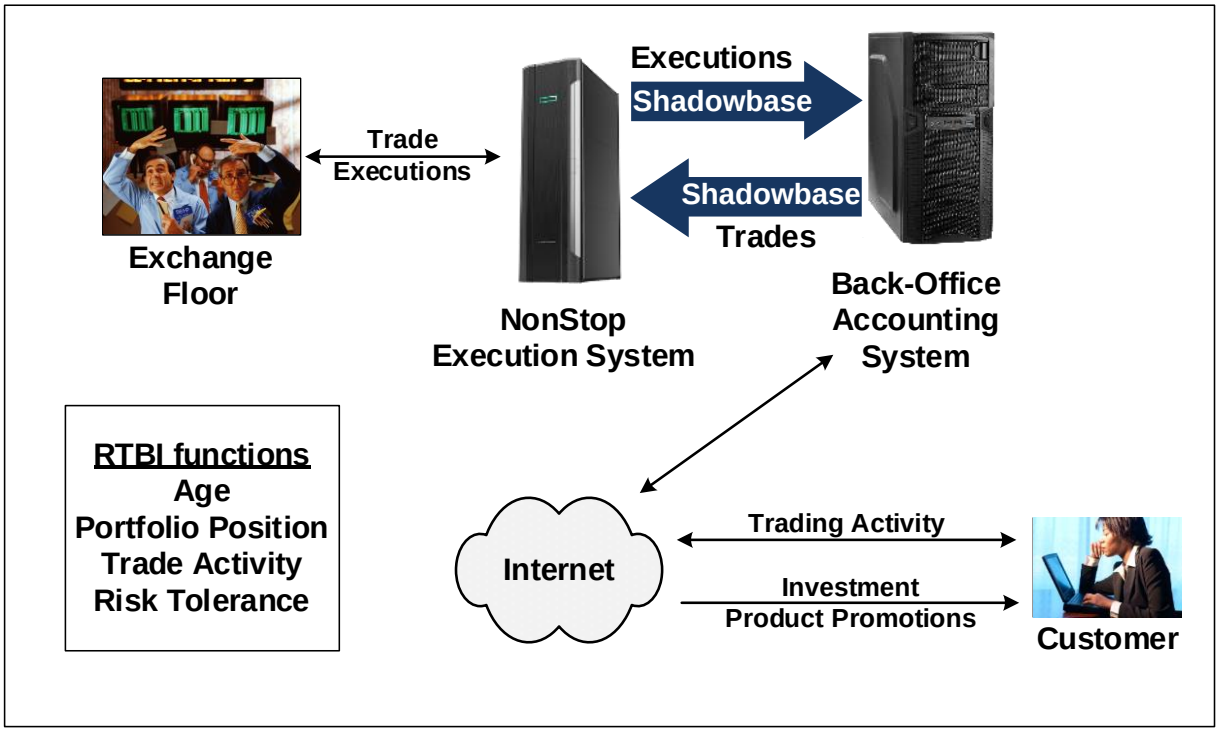


Figure 19 – Investment Product Promotion

A provider of stock trade execution services provides RTBI systems to its brokerage customers. The brokerages use this RTBI system to monitor the trading activity of their individual customers to determine possible investment products in which their customers might be interested.

The execution service provider uses an HPE NonStop system to execute trades submitted to it by its brokerage customers (Figure 19). It sends customer trade requests to the appropriate exchange floor for execution and returns execution reports to the brokerage firm. Connections between the brokerage firm system and the execution system are provided by the HPE Shadowbase data replication engine.

The execution service provider supplies a back office system to its brokerage firms to handle trade accounting and position management. This system also has an RTBI component for monitoring customer activity. Individual customers enter their trade requests via the Internet into the brokerage firm's back office accounting system. Each trade request is forwarded to the execution service provider for execution. The execution report is returned to the customer by the brokerage firm and is used to update the customer's position and trading activity in the back office accounting system.

The RTBI subsystem resident in the back office system continually monitors the customer's portfolio position and trading activity. It analyzes the risk tolerance exhibited by the customer through his activity and, along with the customer's age and other information, determines if investment products provided by the brokerage firm might be of interest to the customer. If so, information on those products is sent to the customer while he is online. This case study is an example of an RTBI system using real-time events to determine potential marketing opportunities, thereby increasing revenues.

Tactical Reordering and Strategic Marketing

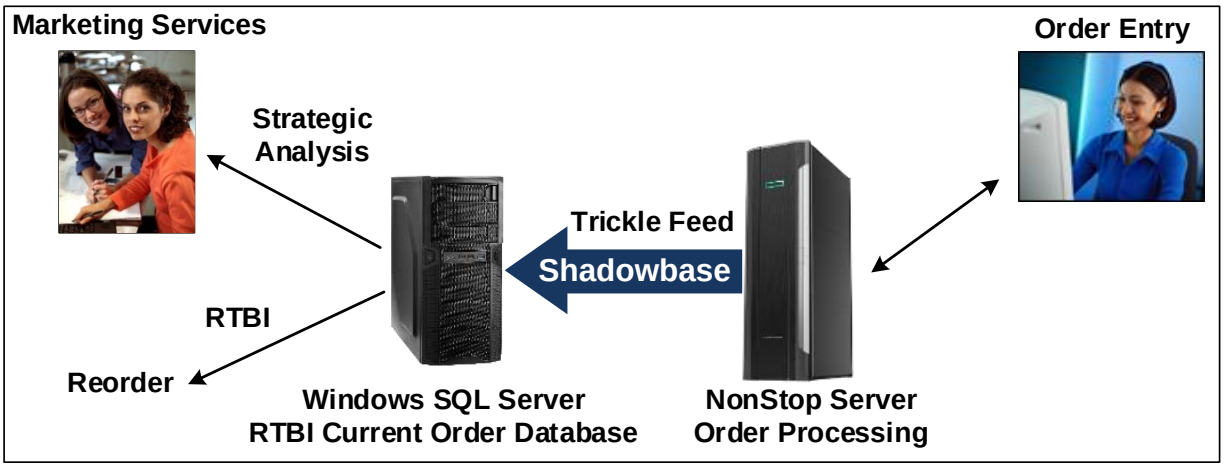


Figure 20 – Real-Time Business Intelligence for Parts Distribution

A major North American independent distributor represents more than three million parts critical to the operations of *maintenance, repair, and overhaul* (MRO) and *original equipment manufacturer* (OEM) customers in virtually every industry. It also provides engineering, design, and systems integration for industrial and fluid-power applications as well as customized mechanical, fabricated rubber and fluid-power shop services.

Its network encompasses more than 4,600 associates at 445 facilities in the continental United States, five Canadian provinces, Puerto Rico, and Mexico. Its seven strategically located distribution centers warehouse millions of dollars of inventory to support its local service centers and customers with necessary products. The distributor uses an HPE NonStop server to accept orders from its customers and to fulfill these orders (Figure 20).

To better improve operations, the distributor implemented a RTBI system running under Windows SQL Server. This system is driven by order information from the HPE NonStop order-entry system delivered via an HPE Shadowbase data replication engine. The RTBI system maintains a database of order history and current orders.

Using the order history and analytic strategic processing, the distributor determines what marketing services will be the most effective to grow its business (a strategic data warehouse function). Using current order activity, it determines in real-time when parts need to be reordered (a tactical RTBI function). This case study is an example of feeding an RTBI system that supports both strategic analytics and tactical responses to current events in order to increase revenues and improve customer satisfaction.

Travel Analysis and Profiling

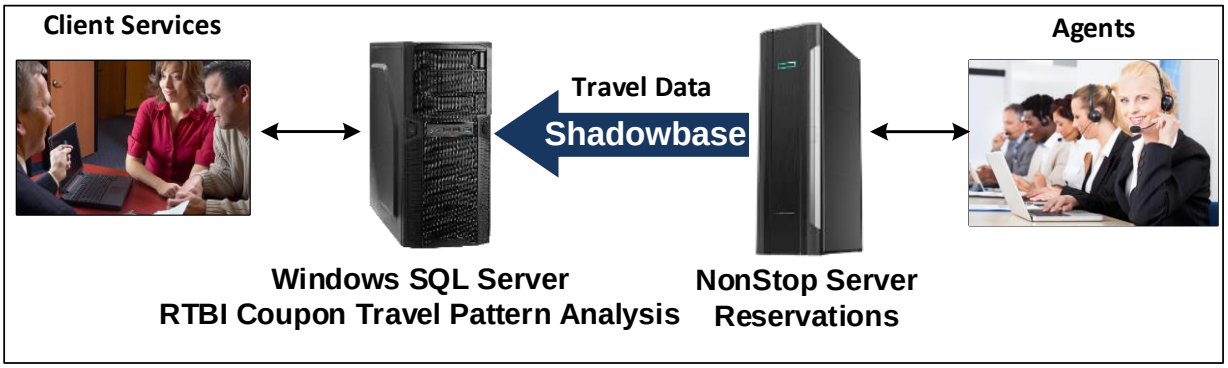


Figure 21 – Real-Time Business Intelligence for Travel Packaging

One of Europe's leading providers of tours and packaged travel offers an expansive array of flights, hotels, package tours, flexi-packages and independent holidays to travel agencies and several million consumers throughout Europe. Its travel brand offers flexible and fixed vacation packages to major leisure destinations worldwide. It supports hundreds of travel agents with an HPE NonStop server system. One of the provider's services is to offer discount coupons to its clients for frequently traveled routes and destinations. To determine the applicability of coupons for various itineraries, it uses a RTBI system to track travel patterns and volumes.

This business intelligence system runs on a Windows SQL Server (Figure 21) and uses online analytical processing to discover travel patterns of potential candidates for coupons. Travel data from its NonStop server is sent in real-time via the Shadowbase data replication engine to the business intelligence system as the provider's travel agents book travel and accommodations for their clients. Using this architecture, the company determines in near real-time appropriate travel patterns, and then offers timely travel and leisure discount coupons to its clientele, for increasing revenue generation.

Summary

The speed of today's processing systems has moved classical data warehousing into the realm of real-time. The result is real-time business intelligence. As business transactions occur, they are fed to a RTBI system that maintains the current state of the enterprise. The RTBI system not only supports the classic strategic functions of data warehousing for deriving information and knowledge from past enterprise activity, but also provides real-time tactical support to drive enterprise actions that react immediately to events as they occur (real-time decision support). It replaces both the classic data warehouse and the enterprise application integration functions. Such event-driven processing is a basic tenet of RTBI.

The HPE Shadowbase suite of products provides the capabilities necessary to integrate disparate operational application information into the RTBI system and to support RTBI functions. The HPE Shadowbase replication engine is a high-speed, bi-directional, heterogeneous data replication engine that moves data updates from enterprise systems to the RTBI system and back in fractions of a second. It allows the events to be filtered and transformed into formats that downstream processing can utilize, and embeds complex processing rules into the replication engine instead of having to encode them into the applications themselves. It supports active/active configurations for RTBI systems to achieve continuous availability.

With RTBI, an enterprise establishes long-term strategies to optimize its operations, while reacting with intelligence to events as they occur. HPE Shadowbase software provides companies the suite of products necessary to leverage this RTBI technology with proven implementations, while providing significant business benefits.

International Partner Information

Global

Hewlett Packard Enterprise
6280 America Center Drive
San Jose, CA 95002
USA
Tel: +1.800.607.3567
www.hpe.com

Japan

High Availability Systems Co. Ltd
MS Shibaura Bldg.
4-13-23 Shibaura
Minato-ku, Tokyo 108-0023
Japan
Tel: +81 3 5730 8870
Fax: +81 3 5730 8629
www.ha-sys.co.jp

Gravic, Inc. Contact Information

17 General Warren Blvd.
Malvern, PA 19355-1245
USA
Tel: +1.610.647.6250
Fax: +1.610.647.7958
www.shadowbasesoftware.com/
Email Sales: shadowbase@gravic.com
Email Support: sbsupport@gravic.com

Hewlett Packard Enterprise Business Partner Information

Hewlett Packard Enterprise directly sells and supports Shadowbase Solutions under the name ***HPE Shadowbase***. For more information, please contact your local HPE account team or visit our website.