

Addressing BI transactional flows in the real-time enterprise using GoldenGate TDM

Alok Pareek
Vice-President, Technology
apareek@goldengate.com

Abstract

It's time to visit low latency and reliable real-time infrastructures to support next generation Business Intelligence (BI) applications instead of continually debating the need and notion of real-time. The last few years have illuminated some key paradigms affecting data management in operational systems. The arguments put forth to move away from traditional DBMS architectures have proven persuasive - and specialized architectural data stores are being widely adopted in the industry[*Stone et al.*]. The change from traditional database pull methods towards intelligent routing and push models for information is underway, causing many applications to be redesigned, redeployed, and re-architected. One direct result of this is that despite original warnings about replication [*Gray et al.*] – enterprises continue to deploy multiple replicas to support both performance, and high availability of real-time applications, with an added complexity around manageability of heterogeneous computing systems. The enterprise is overflowing with data streams that require instantaneous processing and integration, to deliver faster visibility and invoke conjoined actions for real-time decision making, resulting in deployment of advanced BI applications as can be seen by stream processing over real-time feeds from operational systems for complex event processing (CEP) [*AI_n27385199*]. Given these various paradigms, a multitude of new challenges and requirements have emerged, thereby necessitating different approaches to management of real-time applications in the operational BI area.

The purpose of this paper is to offer an industry viewpoint on how real-time affects critical operational applications, evolves the weight of non-critical applications, and pressurizes availability, and data movement requirements in the underlying infrastructure. I will discuss how the GoldenGate Transactional Data Management (TDM) platform is being deployed within the real-time enterprise to manage some of these challenges particularly around real-time dissemination of transactional data across multiple systems to reduce latency in data integration flows to enable real-time reporting, deploy real-time data warehousing, and to increase availability of underlying operational systems. Real world case studies will be used to support the various discussion points. The paper is an argument to augment traditional data integration flows with a real-time technology (referred to as *transactional data management*) to support operational BI requirements.

Key words: Real-time, Business Intelligence, Change Data Capture, Replication, Reporting, Data Warehousing, Transactional Data Management, ETL, Data Integration

Submission Category: Industry Track

1. Background

Enterprises have been deploying applications for strategic business intelligence (BI) purposes for some time now. A significant problem with modern BI deployments is the challenge associated with reducing the latency of the data that the BI applications operate on. Data freshness is not much of an issue where BI applications are used for longer term strategic decision making. Currently, most BI applications with real-

time information either need to resort to querying an operational system, or are deployed against a recent point-in-time (snapshot) copy obtained via clone versions created using business continuity volumes (BCV), or finally through classic ETL techniques. Associated with these two approaches are operational *impact* (performance, application modification), *cost* (supporting expensive staging areas, multiple disk subsystems, and IT resources), *complexity* such as maintenance of ETL workflows with shrinking batch windows, multiple ways of data extraction, ad-hoc responses to failure scenarios during traditional ETL workflows, and data staleness due to periodic batch ETL. Many of these challenges, and problems have been recently described by researchers – the Qox proposal [Dayal *et. al*] talks of an approach to systematically describe quality objectives for, and generalize quality of service, quality of data, quality of information etc. for *performance, reliability, maintainability, freshness, scalability, availability, flexibility, robustness, affordability, auditability, and traceability* in data integration workflows with next generation BI architectures in mind.

Over the last few years, global companies producing very large data volumes like Paypal, DirecTV, AT&T, and Visa have recognized such complexity and limitations inherent in their current implementations as a hindrance to achieving their real-time goals. In an effort to expedite informational flows across their enterprises, and address many of these quality objectives, they have deployed GoldenGate as part of their real-time infrastructure to meet their real-time requirements in a scaleable and reliable manner. The core technology originated from lazy transactional replication of HP NSK TMF audit logs to enable high availability of ATM switches, and has evolved into *Transactional Data Management* – a way to address real-time propagation of transactional data to support high availability, and real-time continuous data integration across heterogeneous systems and technology stacks.

The key problem areas within the domain of real-time BI that are being addressed by the TDM platform are:

1. Live Reporting
2. Real-Time Data Warehousing
3. Real-Time feeds into Operational Data Stores
4. Integration with Messaging systems
5. Micro-Batching
6. Real-time replicas for HA (to support real-time service level agreements)

2. Operational Data, and the Real-Time Enterprise

Transactional Data

Members of this community hardly qualify for a screed on transactions! But my primary focus in this paper will be on real-time data acquisition, and continuous integration of the class of data that emanates from transactional systems. So, first it's important to confine the context to data originating from transactional systems, and second justify why this kind of data can be treated separately than non-transactional data.

Transaction processing database systems have long served as the backend layer for persistence for a majority of critical business applications. The need for externally sharing (most often by propagating) information written to the database is hardly new, with a plethora of existing technologies allowing data movement and integration with downstream enterprise systems such as operational data systems, data warehouses, specialized data appliances, vertical data marts, middleware based messaging systems, and additional business applications. Popular technologies utilized include *Extract Transform Load (ETL)*, Native database replication, *Change Data Capture (CDC)*, Materialized views (snapshots), Message Buses, and other load/unload utilities provided by commercial vendors specializing in data integration companies providing ETL/EAI tools, relational databases, or custom developed application code.

The reasons for the real-time data sharing are numerous, a majority of them well-understood, such as applications serving critical operational processes requiring immediate analytic insight, intelligent

performance indicators, advanced detection of interesting events, gaining historical data access for auditing in real-time, abiding by continual compliance for regulatory purposes, enabling event driven architectures for addressing time sensitive opportunities, deploying web services etc. There's an emergence of newer applications that offer rich real-time functionality like integrating GPS data with Customer Relationship Management (CRM) systems, or data from mashups to supplement real estate data applications with cartographic data. Lastly there are newer applications that will be built on advanced data accessibility, and management frameworks such as dataspace. [Michael et. al] that may require storage, or event propagation from heterogeneous participants.

Given the deep obsession by enterprise IT departments with performance of operational databases, the amount of high overhead tolerated in interactions with other systems is surprising. Database kernel developers write a fair amount of highly optimized kernel code to ensure high transactional throughput, streamlined execution paths to shorten transaction commit times, ACID compliance, and high availability. However, the interaction of many of the aforementioned information sharing technologies with the underlying transactional database ends up degrading throughput, disrupting cache patterns (for example by negatively influencing LRU page replacement victims), breaking up ACID guarantees when transferring the data sets even though otherwise desirable to the downstream subscribing systems, and reducing availability of the underlying database by increasing query response times. These performance issues impose a challenge in large scale applications towards providing predictable, real-time responses.

Large mission critical businesses are increasingly beginning to pay attention to this oddity that's being rendered transparent due to new emerging trends in data volumes, large number of users, advances in processing technology, ubiquitous connectivity, and altered user expectations. I will briefly describe these trends, but first let's get past *real-time*.

Real-Time

Definition

It is challenging to get solid consensus on the precise definition of *real-time* particularly across IT managers, researchers and system implementers. Effective arguments have been presented on the concepts of real-time, near real-time, just in time, right-time etc. as they apply to business processes, analysis, operational decision making, and addressing information delays in the Real-Time Enterprise (RTE) environment. The following figure illustrates the point using data from respondents in a survey conducted by TDWI. [TDWI_3008].

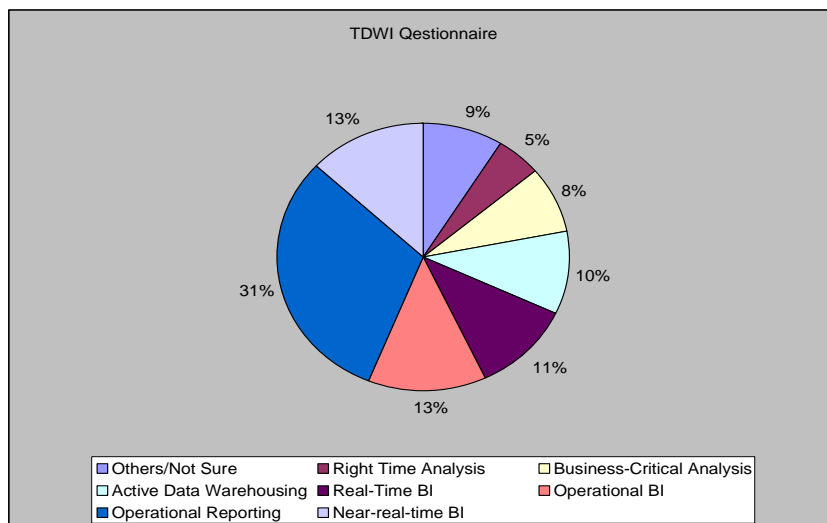


Figure 1. TDWI Survey: Trends in Operational BI. *Which Term does your group use?*

Real-Time Interval

The focus in this paper is primarily on mission critical transactional data that needs to be processed across multiple systems in the RTE. As such, the definition of real-time is proposed in terms of an interval: for a given data item X , a real-time interval RT_1 is taken to be the wall clock interval between the data item's birth time X_b and its visibility time X_v to any requesting service, an end user, or an interested application. In most relational database systems, where X is a tuple in some relation R , we take X_b to be the logical commit time or commit sequence number of the most recent change that modified X (implemented using either logical or physical clocks by the RDBMS), and X_v for that same tuple to be the time at which the causal results induced by the tuple's modification are queryable. Clearly, RT_1 is close to zero when X is queried in the same system containing R . (In this definition, RT_1 could possibly be negative since in many implementations, the uncommitted tuple is subsequently visible to the session creating X prior to X_b . However, this data is still uncommitted. In our context RT_1 can be treated as zero for this case).

RT_1 for X becomes interesting only in the visibility context of non-originating systems of X i.e. other systems that X (or derived values from attributes of X) is either propagated to, or queried from. The main reason for these definitions is to clearly demarcate the boundary between when a system *should* see X , and when it *can* see X . Many corporate applications seem to a priori derive their service level agreements (SLA) requirements factoring in limits inherent in traditional data movement technologies (most often some flavor of ETL), but the SLA determination should be based on business requirements rather than technology capabilities. With the availability of technologies that can satisfy RT_1 in sub-seconds, seconds or even minutes, enterprise analytical business applications have a far greater degree of flexibility in making their visibility selection. We find that many business users desire up to the minute information, but are being delivered information on a daily or hourly basis because of the high impact to the operational system due to the data propagation overhead of traditional data movement methods. In fact, to correctly satisfy real-time requirements, the visibility time X_v should be governed by any latency inherent in the underlying propagation technology, whereas, the question of when X should be used by an application needs to be a business decision.

A key problem in BI applications for tactical decision making is satisfying a narrow RT_1 for transactional propagation such that any BI related lookup or computation can be achieved within that interval with a feedback loop to the application that created the transaction which triggered the BI processing. Synchronous writes from the source systems to the BI system is a way to keep RT_1 close to zero – but this obviously doesn't scale because of classic bottlenecks introduced by some variant of a two-phase transaction commit protocol. Therefore, an asynchronous high-speed transactional data acquisition and propagation method must be deployed to support the real-time interval for meeting an enterprise's real-time requirements.

Heterogeneous Systems and Interoperability

Mission critical applications are being deployed today across a variety of databases, appliances, operating systems, and hardware platforms. In the emerging real-time BI area, the landscape looks as follows - first, we see a range of operational systems deployed from legacy mainframes, to relational databases coming from a variety of different vendors like Oracle, MySQL, Sybase, DB2, Microsoft SQL Server etc. Major enterprises have hundreds to thousands of these system deployed with a chaotic flow of data between them, often in an ad-hoc fashion. (I must add that there's usually some data architect who can readily present a historical justification for any given data flow edge between a pair of database nodes). Reducing complexity in these data flows is seen as a significant challenge. Many IT projects related to data consolidation, re-hosting, or datacenter migrations are a way to simplify this complexity, and a single system that can be queried for a real-time view of the business is required. Individual business units are reluctant to redeploy their applications unless there's some way to fallback on the conventional way to retrieve application data. This means that the two systems have to operate together with real-time data

feeds in a bi-directional manner until sufficient application testing is conducted, and one system is phased out.

Next, we see eager exploration, and accelerating deployment of newer systems often using commodity hardware that are exclusively catered for specific data warehousing (DW), and BI applications. These systems require real-time data from a variety of different data sources. It is simpler to have one way to address data acquisition into the system, rather than implement a host of individual data connectors from the source systems. Another challenge with the newer systems is that due to their specialized functionality, the interfaces to integrate real-time data may be limited, or require application development efforts.

Finally, there appears to be a resurgence of interest in in-memory databases and data grids being used as subscribers of real-time data for visualization, dashboard implementations, business activity monitoring (BAM) etc. There needs to be a way to move data of interest from the BI system into these systems.

Transactional Consistency

To support operational BI, data feeds into operational data stores, or an enterprise data warehouse do not generally require complex transformation, or aggregation. However, the data does need to preserve transactional consistency since online applications are continuously querying the stream of incoming data to make up to the minute decisions for data driven feedback. This consistency refers to satisfying the constraints specified at a schema level in addition to additional database level constraints. Many of the current technologies used to support BI information flows from operational systems capture data at a table level, often using application generated timestamps or logical sequences, thereby not preserving all the constraints (for e.g. referential integrity) or necessary transactional metadata (for example multiple operations that were submitted as part of one database transaction). As a consequence, the loads into the BI systems are done in an offline mode so that eventual consistency can be restored upon completion of the load. This works well for traditional data warehousing but is a serious problem in real-time applications since there are no batch windows in real-time environments for offline processing.

3. Emerging Trends and Problems

Amount of Data

The amount of human and machine generated data being created and with processing requirements within a narrow ensuing window of its birth is growing at an unprecedented scale. As an example of a real world data point from the Utilities sector - current supervisory and data acquisition systems can publish data about the state of transmission lines once every 4 seconds. With the advent of smart grids, synchrophasors can sample the voltage and current measurements to 30 times a second [*Econ_2009*] – for a smart grid BI application that needs to react to real-time fluctuations – the amount of data that requires processing for real-time decision making such as early detection, and prevention of massive blackouts just grew by 11900%. The response to the increased data volumes by companies such as the New York Independent State Operator (NYISO), Xcel Energy, PJM and others has been to explore fast, reliable real-time data acquisition within their existing infrastructures for enabling real-time data integration.

To put some real world numbers behind the volumes from a cross section of GoldenGate customers in the Banking and Finance sector, online telecommunications, e-business, and travel, transaction log generation rates of 500G to 2 TB a day are common. Typically this translates to thousands of transactions per second on a single system. The data from these transactions serves as events to downstream analytic systems, maintaining a real-time latency across the operational and analytic systems in the face of such volumes is a major challenge.

Adoption of newer datatypes

In addition to the amount of data, sizes of individual data items are also growing. With increased adoption of newer datatypes such as native XMLTYPE for storing unstructured, and hybrid data, DICOM as a standard for Digital Imaging and Communications in Medicine, RDF to support semantic web technologies etc. in the industry, a significant amount of semi structured and unstructured data is being stored, queried, and manipulated within relational databases. Existing structured records are widely being augmented with image, audio and video data. Modifications to these records need to be efficiently propagated across multiple systems rather than relying on traditional means to move them in their entirety. This has impact on replication, ETL, CDC infrastructures, and many logging optimizations in the database when writing large data sets to the database.

Growing number of Users

The number of information consumers on enterprise systems has grown considerably due to a number of enterprises offering interactive, consumer facing services from a web front. This has created scalability and performance challenges because the system behavior has become less predictable due to data access from multiple consumers, often in an uncontrolled manner. An interesting example is the case of Sabre Holdings, an online travel reservation system that had to accommodate millions of new online users (mainly browsing for inexpensive fares) thereby skewing the query to write ratio by orders of magnitude as consumers searched and compared numerous travel options before making a purchase with an optimal fare. Scaling existing infrastructures is cost prohibitive to accommodate such growth, thereby seeking novel solutions.

The changing nature of applications

The boundary between mission critical applications, and commonly used applications is fast blurring, and the line is dependent on real-time. Consider for example, a data warehousing system where information is being transferred from several data sources on a nightly basis. The users of an application deployed on this warehousing system are used to working with nightly snapshot data, so the non-availability of the data warehousing system for duration of a few hours does not seriously affect any business processes. However, if the data warehouse is being continuously fed real-time data from the operational systems, users of the warehouse begin to expect and rely on real-time response times from their BI applications. As a direct consequence, the availability requirements on the data warehouse become stringent. An evolution of applications from being non-critical to critical is underway because of such reliance on real-time data.

Another trend similar to operational systems is adoption of dual site data warehousing implementations to address any planned or unplanned outages so as to not disrupt real-time applications.

Micro-Batching

Recent research from Gartner predicts that by 2010, more than 90 percent of all production data warehouses will have intraday or better updates and more than 50 percent will be updated continuously. Additionally, according to a recent IDC end user survey, over 70 percent of respondents said they will be acquiring real-time data within 12 months for their business intelligence environments. In contrast, 30 percent of respondents are acquiring data in real-time today to support business intelligence activities. [Brobst et. al]. However, most traditional data warehouses are refreshed in an offline manner, and periodically – on a nightly, weekly, or monthly basis. ETL, ELT processes are used to extract the changed data set, undergo data cleansing and transformation, and integrated the data into the data warehouse within a processing window known as a batch window. In the internet era, major businesses have worldwide users, and need to operate their systems on a 24x7 basis - no longer having the luxury of the traditional batch window. There's a strong desire for businesses to offer real-time visibility, and proactive responses to business users within their organization, and real-time notifications, real-time interactions, and real-time responses to and from their consumers externally, this is causing system degradation and putting pressure on system architects to either update their warehouses on a real-time basis or speed up processing windows to reduce latencies in data flows across their enterprise. We are beginning to see smaller batches with

update intervals of 15 minutes up to an hour. Contrasted to nightly batches, this sort of batch update is referred to as *micro-batching*.

4. Real-Time data acquisition

Real-Time data capture (acquisition) is the first step in meeting the real-time requirements for operational BI. The prevalent technologies used in data capture in a majority of organizations today are ETL, and CDC. As mentioned earlier, the major problems with real-time data movement between operational systems using these conventional data movement methods are shrinking processing windows, and vast expansion in data volumes that introduce a significant lag in the real-time BI pipeline. Both these technologies need to evolve to support real-time.

ETL and real-time challenges

The most common method used by organizations to deploy BI systems is to use commercial or custom ETL to extract, cleanse, transform, and load data into downstream systems. This is a mature, well understood technology and has been discussed at length in literature [InmonDW]. In real-time environments, ETL introduces significant latency and performance overhead in operational systems. The high latency arises due to ETL's batch oriented approach, and operational cycles consumed by extracting data sets from operational tables, typically by scanning application tables. The mere fact that ETL vendors' benchmarks emphasize numbers for fast load times [*SSIS_ITB*] ignores the latency component resulting from the extract step.

Even in ETL implementations using CDC to optimize data capture, queue maintenance to support the CDC infrastructure within the operational system adds significantly to the latency, and additionally, has an adverse effect on operational throughput. The next section provides supporting benchmarking data when database change tables are used for ETL queues.

Another challenge with ETL is that to satisfy real-time requirements, BI applications need to continuously query to react to the incoming data that is being applied in a continuous manner at the target system. Since ETL loads are optimized for batches, they do not preserve consistency during the load - the target system therefore cannot be consistently read to support real-time BI applications.

Data acquisition into a warehouse is also affected by any failures during the ETL flow. The impact of such failures becomes magnified in a real-time enterprise, where the service levels for data freshness are critical. If the ETL execution crashes during the extract, transformation, or load processes, it is difficult to meet the time frames for batch ETL processing unless highly granular checkpoint and restart capabilities are built into the implementation. In this respect, smaller batches are better for managing catch-up time when reprocessing is required.

ESB

Enterprise Service Bus (ESB) implementations do allow acquisition of data in real-time. Organizations that can publish transactions onto the ESB can have the downstream systems subscribe to the data thereby achieving real-time requirements. However, most organizations do not seem to have an efficient, scalable ESB architecture in place. In enterprises deploying a service bus to share data in a publish/subscribe manner - the common method for delivering data to a messaging system is to alter the participating applications with custom built or EAI vendor provided adapters. However, in addition to adding complexity and effort to the messaging infrastructure, this method adds strain on the underlying operational databases, and adds latency with the increase in the number of transactions on the underlying applications.

Change Data Capture (CDC)

CDC has been in use for over a decade. An ideal CDC implementation will capture events processed by an OLTP system without impacting the performance of that system and without requiring application code changes. CDC technologies have historically been used to replicate data across multiple database instances and multiple sites to facilitate disaster recovery and to provide increased levels of availability during planned and unplanned outages. However, clever data warehouse architects have begun to use this technology to replicate data on a continuous basis to facilitate data capture of all events from one or many OLTP systems into a staging area in the data warehouse environment for transformation and loading into the target tables for the warehouse.

Most CDC implementations utilize one of the following four approaches:

1. Trigger/application-based CDC

Using this method, changes are tracked in separate tables directly by the process modifying the data record, or indirectly via triggers in a set of additional tables that can minimally record the key and time stamp/version of the modified record, or complete information to store before images of records. This obviously adds significant overhead to the source operational system.

2. Audit-based CDC

Application tables are augmented with additional columns that, upon the application of data manipulation language (DML) operations against the records in the operational table, are populated with time stamps, change tracking version numbers, or both. The time stamps allow the detection of records modified since the last time data was extracted from the table in order to select changed data for the next incremental load. The drawback here is the overhead due to index and table scans to process the next set of data that needs to be extracted.

3. Snapshot-based CDC

In this method, a previous copy of the operational database is compared to the current copy to determine the set of changes. This technique is rarely used because of the growth in operational systems.

4. Log-based CDC

This approach enlists a reader process to monitor a source system's transaction log, which contains information for database recovery. In this approach, the log is usually enhanced to capture additional data and metadata required for target-side change application. This is the most efficient way to monitor for changes without impacting the source system. Several database vendors offer CDC APIs to capture changes within their databases. Efficient continuous integration requires that these changes be persisted in queues so that change propagation can operate independently from change capture. Within homogeneous systems, database vendors often provide their own replication CDC, queuing, and change apply features to move the changes to a downstream system. For example Microsoft Transaction Replication allows for a publish/distribute/subscribe feature to replicate changes to downstream Microsoft SQL Server systems.

For faster refreshes of data warehouses etc. SQL Server 2008 provides a CDC feature that uses a SQL Server Agent job to capture insert, update and delete activity. This information is stored in a relational table, from where it can be accessed and used by data consumers (for example SQL Server 2008 Integration Services SSIS). CDC can then be used in conjunction with SSIS to incrementally populate data warehouses, enabling you to produce more frequent reports that contain up-to-date information [Microsoft_SS2008].

In order to get quantitative evaluation of CDC overhead we decided to measure some key metrics when using CDC. For our testing, we used a Dell PE2950 system running SQL Server 2008 (SP1) on a quad core processor with 2x6 MB Cache, 3.16GHz clock speed, 8Gb RAM, 1333MHz FSB Windows Server 2003 R2 EE SP2 (32bit) configuration – and a non audited TPCC-variant workload (simulating business transactions akin to the TPCC transactions submitted through multiple clients).

The following figure shows the effect on throughput (measured in operations per second), amount of redo overhead, and system CPU.

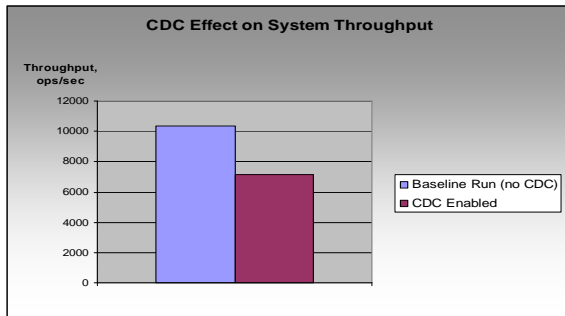


Figure 2: CDC effect on throughput

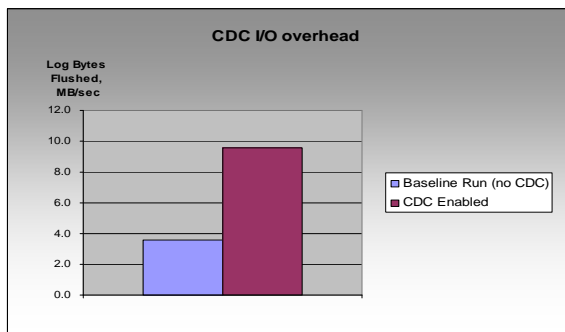


Figure 3: CDC effect on redo overhead

Effect of CDC to track table changes in SQL Server 2008. The three figures measure baseline performance with respect to throughput, additional logging, and CPU used.

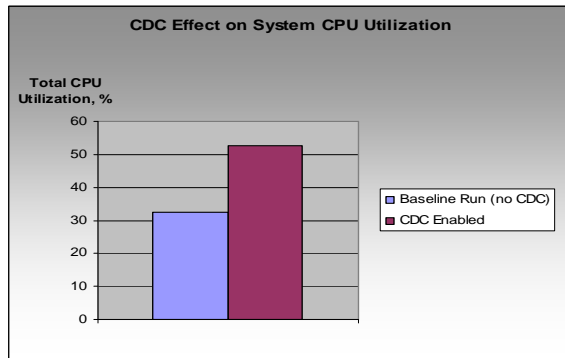


Figure 4: CDC effect on system CPU

As can be seen from the data above, many businesses that generate high volumes of data are understandably reluctant to use native CDC implementations, or are in the process of transitioning to newer ways of acquiring real-time data. It is important to note that deploying CDC is the first step that allows an application to get changed data. How to propagate this downstream to heterogeneous systems, subscribe to the data, and manage failures in the network, manage guaranteed delivery, and optimize for different workloads is a complex undertaking and requires deep expertise in systems programming, replication implementation, and recovery across a disparate set of data stores.

5. GoldenGate TDM Platform

Towards the early part of this decade, Bosworth [Bosworth_2002] pointed out that as the Internet morphs it will result in information going through and forming what he described the InformationBus. A key point in that paper was that the challenges over the next 20 years was interacting with the information bus to support scaling, routing, querying, filtering without causing total system degradation. Noteworthy were calls to focus on push models with emphasis on asynchrony.

Responding to these challenges, we at GoldenGate have built a data management middleware software platform to satisfy real-time infrastructure requirements, focusing on data originating from transactional databases that serve as the backend systems for enterprise operational systems. We refer to this as *Transactional Data Management (TDM)*.

TDM must meet the following three criteria:

1. *Real-time* - this means that the real-time interval RT_1 must be short, usually under a second, or order of seconds based on application expectations. Generally, an RT_1 of minutes does not qualify as real-time in most enterprises where instantaneous visibility of operational data is required to make operational BI decisions.
2. *Heterogeneous* -TDM must support moving transactions across different architectures, operating systems, endian systems, storage, databases, and application versions.
3. *Transactional* -TDM must preserve the ACID properties when moving data across multiple systems.

GoldenGate TDM is already implemented in thousands of mission critical environments worldwide to support their real-time data warehousing applications, real-time reporting applications, and real-time messaging applications. The software works at a tier lower in the software stack than that offered by integration vendors like TIBCO etc. that gets deployed at the application tier, or ETL technologies that typically issue reads on data pages of operational tables. TDM offers a way to address fast, scalable transactional data dissemination without introducing changes in the application logic, or adding overhead introduced by database triggers, or utilizing poorly implemented CDC infrastructures by database vendors that though functional, were not designed with low latency, transactional consistency, or support for very high data volumes in mind.

Critical business services such as Billing, Payment processing, Patient record management, Customer relationship management, POS integration, and Fraud detection are generally implemented as packaged applications. For an enterprise, publishing transactional data from the underlying systems i.e. data generated as a result of database changes made by these applications may not be easily possible unless the packaging vendor offers these interfaces via standards or allows application extensions. Even if such facilities are available, additional development effort is necessary, and could (and frequently does) impact performance. Changes introduced in the application logic are costly both from a resources and time perspective, and often cannot scale with the large number of changes being processed by the underlying database especially when changes requiring propagation might have derived values, and additional metadata that gets integrated from other database objects.

Therefore, a significant advantage in GoldenGate TDM is that application changes are not required for real-time data acquisition making BI transactional flows easier, and faster to implement.

GoldenGate Architecture

The GoldenGate TDM platform enables asynchronous real-time transaction sharing across different systems by implementing a loosely coupled process based architecture that's independent of both application and database layers. Transactions from supported systems can be captured, routed, transformed, delivered and verified across heterogeneous environments in real-time. As an example in the Telecommunications sector, transactions from a high volume billing application running on an Oracle 11g database can be captured, and integrated within real-time in sub-seconds into a Teradata active data warehouse that maintains a single consolidated view of the customer. The captured transactions can additionally be integrated into an Informatica server to feed the ETL workflow to reduce capture side latency.

The view of the platform is both transaction-oriented, and record (tuple) oriented, offering rich manipulation functionality at those granularities in the data capture, data distribution, and data load tiers.

GoldenGate abstracts the flow of operational data by using a proprietary high speed queuing mechanism (referred to as *GoldenGate Trails*) backed by files for persistence. Once the transactions are captured, they are written to the trail in a canonical format – thereby enabling propagation to a variety of heterogeneous systems that include relational databases, data warehouses, data appliances, stream based/CEP systems, messaging systems, XML formats, or third-party applications via C or Java custom exits. The following conceptualizes the flow of transactions (or records) within the enterprise.

The end-to-end flow offers a uniform, standard method to capture, route, transform, filter, and apply transactions between a variety of heterogeneous systems including popular commercial systems like Oracle, Microsoft, DB2 (LUW), DB2 Mainframe, Teradata, Sybase, MySQL, HP NSK Enscribe, SQL/MP.

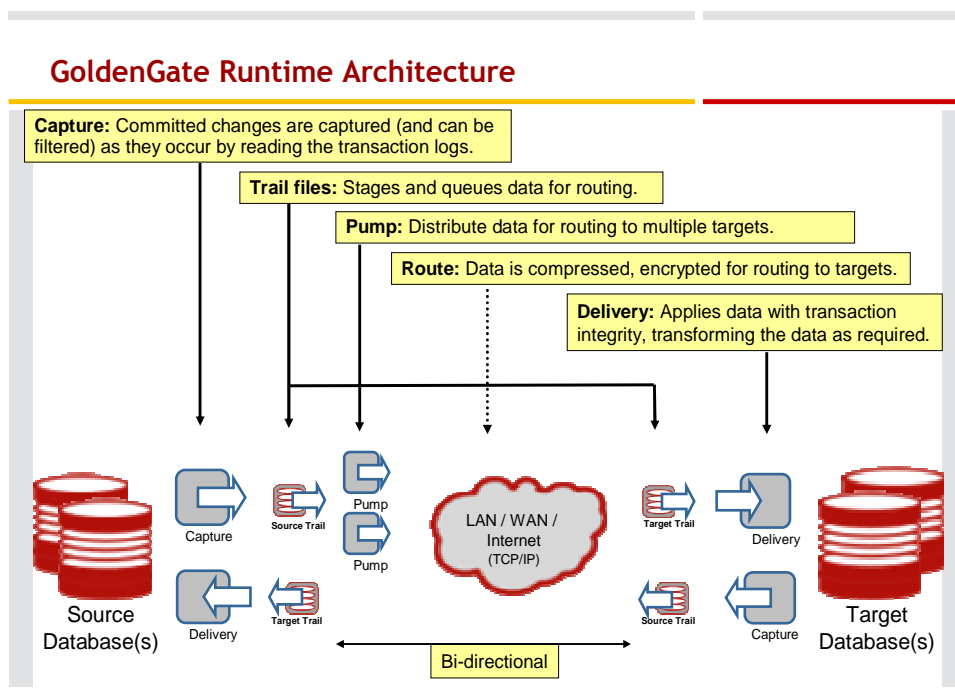


Figure5: GoldenGate process based runtime architecture

Key Components

Capture – Data Extraction

The GoldenGate Capture component is implemented as a threaded stand-alone process that can asynchronously capture committed DML operations, and DDL changes from a variety of data sources. The most commonly deployed topology is a one time *initialization* from the source database's *tables*, followed by continuous capture of changed data (DML operations) from the database's *transaction or redo log*. The synchronization between the initialized first copy of the target system and ongoing changes against the source database is addressed in the software. No application quiescing is required to create a first copy. Because changes are extracted from the log, no operational system impact is introduced.

High speed APIs are used to read the database logs. For certain platforms where the underlying databases may not implement write ahead logging (WAL), a GoldenGate API (VAM – next section) can be implemented to capture changes from system memory/disk with different transaction commit semantics. Data can be filtered, mapped, and converted via configuration at the process tier. Multiple Capture groups can be used to scale the workloads.

VAM API

The VAM is a communication layer that passes data changes and transaction metadata to the Capture process. The GoldenGate VAM API allows for capturing changes from newer data sources via a flexible and extensible pluggable architecture.

The key components when using VAM based capture are:

1. *Capture with VAM API* – a standard GoldenGate Capture process compiled to capture data from a VAM API, then linked with the source VAM implementation.
2. *Source VAM implementation* – this is the implementation that retrieves transactions from online logs and archived logs using the source log reader API.
3. *Source log reading API* – interface to lower level functions that implement the VAM required functionality in a few functions: to initialize, set initial position, read complete records and obtain column values from the log records.

An example implementation of GoldenGate's VAM based Capture is the Change Data Capture facility of the Teradata Database to collect change data from a source server for application to a subscriber server. [Teradata_Replication]. This implementation enables real-time replication of transactions across multiple Teradata databases.

Pump - Data Distribution

GoldenGate Pump is a specialized Capture process that reads GoldenGate trails and can route transactions over WAN, LAN using TCP/ IP. Typically, the amount of data transmitted is a fraction of the transaction logs that are generated by the database since redo metadata, index metadata etc. are not required for transaction propagation. Since only committed transactions are propagated, intermediate activities and rolled-back operations are not transferred. Traffic is optimized by bundling individual records into larger, more efficient packets and avoiding inherent bottlenecks in processing at a record level. Several levels of data compression are available to further reduce the amount of network bandwidth required for transmission. Depending on data types, data compression can reduce byte transfer by 75% or more.

To support real-time integration requirements, a data pump process can be configured to perform standard processing, including data filtering, mapping, and conversion, or in pass-through mode, where data is passively transferred as-is, without manipulation. Pump can additionally encrypt data over the network for security reasons.

GoldenGate Queues – Data Buffering

Early on, we found that utilizing generic database queues to share transactions introduced overhead to the underlying database and added significant latency in the flow of data across multiple systems. Addressing the range of volumes for data propagation using in-database queues also adds overhead to the underlying database because of the increased activity in the database due to additional indexing, logging, and storage requirements. Some database vendors like Oracle have optimized their queue implementation to use buffered queues [Deiter_2002] but for the sort of high volumes that large customers want to support, latency times still far exceed real-time requirements.

Queuing is implemented in GoldenGate using file based persistence. The queue is referred to as a trail and is written by the Capture component using sequential IO, and consumed by either the Pump - the distribution process, or by the Apply process. A trail consists of variable sized transaction records that mainly contain transactional data along with context created by the generating process such as Capture. Trails store both DML, and certain DDL changes captured from a source database.

Trails are read sequentially by one or more Pump or Apply processes. Trail records are written in an efficient proprietary canonical format in order for multiple heterogeneous downstream systems to be able to process these records thereby ensuring a loosely coupled architecture that abstracts trail generators from trail consumers. Trails are generated in ascending sequence order as a series of files. As data is captured, new trail records that contain the logical record images are packed into trail records and appended to the current file in the trail sequence. Metadata at the record level, object level, as well as the source database level can be additionally written as part of the trail record. Once physical EOF is reached, a new file in the sequence is created, and records are appended to that file. The canonical format allows the trails to be processed across disparate systems. For example - data written to the trail by a Capture process reading a transaction log in a Microsoft SQL Server database can be consumed by an Apply process that is writing the records into a DB2 Mainframe, Teradata, Oracle, or MySQL database. Extensions are allowed so that trails can be processed via third party libraries - for example, a capture process could produce a trail containing data from a source database, a custom process could consume the data, transform it, and produce a second trail, and Apply could consume the resulting trail writing the records to a target database.

In case of network failures, trail files continue to buffer changes for continuous data integration without disrupting the application or database workload. Processing resumes from the failure point using the various process checkpoints to guarantee reliable delivery.

Apply - Optimized High-Speed, High-Volume Data Delivery

GoldenGate's Apply (also referred to as "Delivery") process runs on the target system, reads the extracted trail data, and applies the changes from the trails to the target system using the SQL interface. Changes are applied to target tables using native database calls, statement caches, and using local database access. To ensure data and referential integrity, GoldenGate applies data changes in the same order as they were committed to the source database. A variety of techniques to optimize the posting of changed transactions to the target database are used. Transactions can be grouped, and operations batched thereby relaxing commit frequency, and increasing network efficiency, yet preserving ACID guarantees. Database specific optimizations (such as array loads across operations, dynamic statements to optimize SQL execution) and internal caches are utilized to ensure fast execution of identical statements with different bind values. Multiple Apply processes can be used to implement parallel processing and increase throughput at the target system. Apply supports SQL interfaces via native database API's, or native/generic ODBC drivers to platforms such as Teradata, Exadata, SQL Server, MySQL, Ingres, Sybase, HP Neoview, GreenPlum, Netezza, Aster Data, TimesTen database, HP NSK, DB2 Mainframe, DB2 LUW and others. Supporting newer platforms that support ODBC interfaces is fairly straightforward.

Key architectural Features and benefits

Guaranteed Delivery, Transactional Consistency, Recovery

GoldenGate checkpoints the last changed transaction whenever a commit boundary is encountered. This guarantees the delivery of all committed records to the target, even in the event of process restart, network outage/recovery, or cluster failover. Checkpoints work with inter-process acknowledgments to prevent messages from being lost in the network. Several checkpoints are implemented to persistent media that store pointers to the low water mark of the oldest active transaction, current log position (as a physical byte address), and on-disk write position of the most recent write to the trail, along with a target side database checkpoint corresponding to the most recent commit processed by the RDBMS for a transaction issued by GoldenGate. All processes maintain their own checkpoints.

Transactions are applied in source commit order unless the commit serialization configuration setting is relaxed to allow for parallel processing on very high throughput systems.

Table, Row and Column Selectivity

Record level filtering based on rules can be applied in the Capture, Propagation, or Delivery processes. Records in the transaction log that do not meet the filtering criteria can be discarded. Optional configuration allows for selection and application of scalar transformation rules to specific columns through built-in GoldenGate functions, custom extensions via C or Java callouts, and database level stored procedures. This can be used to reference additional data prior to record processing at the source, or the target system.

Column Mapping and Transformation

Similar to the Capture process, GoldenGate Apply can be configured via user-defined rules to specify selection criteria at a table, column, or data set (row) level. Column mapping and conversion can be performed on the source system, on the target system, or on an intermediary system. By default, GoldenGate Delivery automatically maps any target table column with data from a source table column if they share the same name (*implicit* mapping). Explicit mapping and transformation rules can also be defined, ranging from simple column assignments to more complex transformations for which GoldenGate provides a range of date, math, string, and utility operators. Implicit and explicit mappings rules can be combined. If additional transformations, data quality checks, aggregation or other functionality is required, there are extensions to invoke database side stored procedures, or invoke custom code in the transactional flow.

Conversion of Operation Types

SQL operations can be converted across systems. For example, to maintain an audit log, it would be beneficial to convert source side Update, or Delete operations into Inserts at the target along with additional metadata such as the operation type and possibly a commit timestamp. This can automatically be done using configuration parameters. Many databases store only compressed deletes, and compressed updates in the transaction log for recovery purposes (i.e. not the full row image) – in order to support this use case, special functionality is provided to retrieve the unchanged images using supplemental logging techniques provided by the underlying database, or functionality to update a column to itself to generate the changed data in the log.

Metadata Handling

Metadata across heterogeneous systems is generated using a GoldenGate utility that produces a GoldenGate catalog file that includes information about the format of data that is being propagated, such as the table names, column names, data types, data lengths, and offsets. A catalog file enables GoldenGate to convert data from one format to another when moving data between different kinds of databases. To perform conversions, the definitions of both sets of data must be known to GoldenGate. GoldenGate can query whichever database is local to get one set of definitions, but must rely on a catalog file to get the other set of definitions.

To perform column mapping or transformation on the source system, a target-catalog file that contains the definitions of the target tables. This file is read by the Capture, or Pump process in addition to the source side catalog to perform the conversions.

Non-Intrusive to applications

Because GoldenGate reads changed data directly from the database transaction logs, and requires no changes at the application level, is a non-intrusive solution for delivering data to downstream system. This

evolves the discussion from how to transmit data and focus on strategic business requirements such as which lines of businesses within the enterprise require real-time information. Similarly, the non-intrusive nature of GoldenGate's solution allows application developers to emphasize on functional requirements rather than address infrastructure requirements for high speed data distribution.

Flexible Topology Support

GoldenGate supports a wide variety of topologies. These include one-to one, one-to-many, many-to-one, and many-to-many — with unidirectional or bi-directional configuration support. For scalability, cascading topologies can be created to eliminate any potential bottlenecks. By staging specific sets of database changes on the source or target system, different TDM requirements can be met through a single capture pass on the data source. Each set of staged data can contain unique or overlapping sets of data.

6. Use Cases

The following use cases can be supported using the GoldenGate Platform.

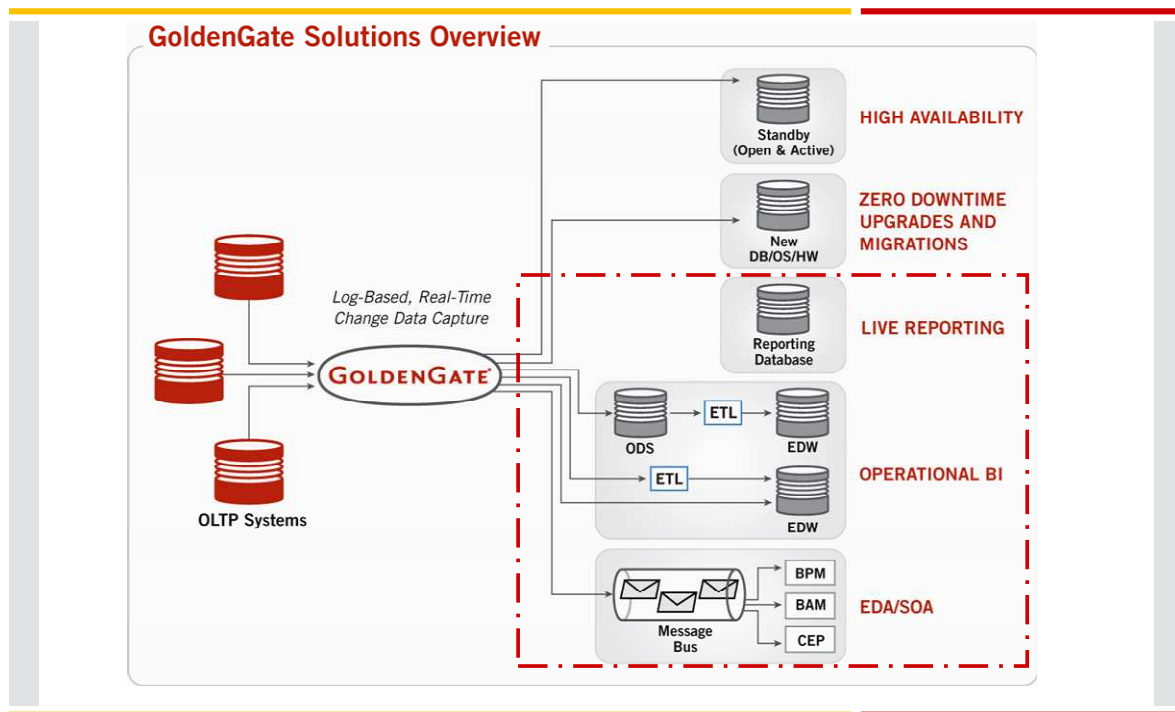


Figure 6: Solutions supported by the GoldenGate TDM platform

1. *Live Reporting*
Off-loading read-only, reporting and analytical activity from OLTP systems to reporting databases. This is commonly used for real-time reporting from packaged applications.
2. *Real-time data warehousing, and Operational Data Stores*

Continuous data integration from multiple source systems into a data warehouse, or ODS with low latency, scalar transformations, and real-time apply thereby eliminating batch windows.

3. *Micro-Batching*

This solution complements existing ETL products with real-time data feeds and eliminates batch windows. It leverages real-time data capture technology to provide real-time data to ETL products with minimal impact on production systems.

4. *Integration with Messaging systems*

This solution enables real-time integration with messaging systems – Changes are propagated via JMS to support event driven architectures, and service oriented architectures.


5. *High Availability*

This solution enables real-time applications to be deployed at multiple active systems with transactional data being synchronized across the systems in real-time. It allows for system scalability and fast failover in case of system outages. Conflicts across multi-master active databases can be detected and conflict resolution methods invoked.

7. Customer Case Studies

Live Reporting, Real-Time Data Warehousing

Case Study: DIRECTV
Real-Time Business Intelligence to Improve Customer Service



Business Challenges:

- Offload Transactions from Siebel system to Active Data Warehouse
- **Corporate-wide SLA:** Perform analysis and decision support on data that is 15 minutes "old."
- Handle growing data volumes: **150 – 200 million records per day** into the data warehouse
- Improve responsiveness of field technicians by optimizing call routes
- Support **major hardware migration** without application downtime.

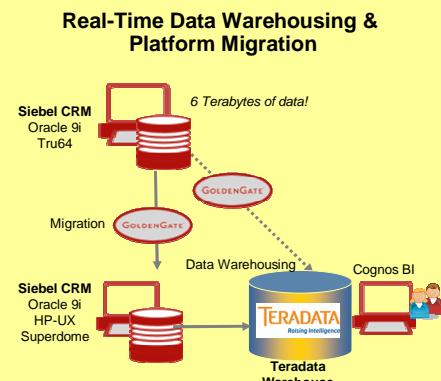
GoldenGate Solution:

- Off-load real-time data from production Siebel/Oracle to Teradata V2R6
- Multi-database, application and platform support

Results:

- Moves Siebel data to the warehouse with **latency of only 1.5 seconds**


Real-Time Data Warehousing & Platform Migration



"With GoldenGate... we know exactly what's happening with our customers and staff across all service areas at all times."
 - Mike Benson, CIO, DIRECTV

Telco / Cable

CONFIDENTIAL - DO NOT DISTRIBUTE



Micro-batching, Reporting

Case Study: Visa

High Availability for Debit Cards & Improved Reporting/BI



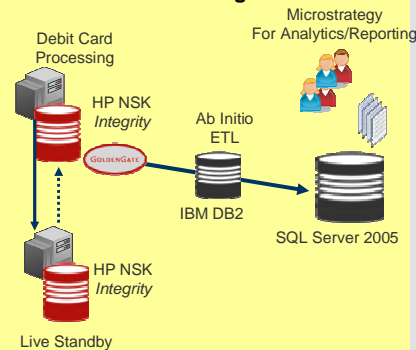
Business Challenges:

- Ability to **maintain data integrity** with **high volumes** of data (over 1 billion transactions p/mth)
- Reduce downtime significantly** especially for unplanned outages (maintenance etc.)
- Near Real-time** continuous feed from HP NSK to MS SQL Server data warehouse for improved BI, reporting.
- Migrate data center** from Washington DC to Virginia with minimal outage.
- Migrate HP NSK System to new hardware platform – HP Integrity

GoldenGate Solution:

- GoldenGate live standby** on HP NSK – reducing overall downtime from **21 hours to under 5 minutes** for fail-over
- Successful **integration of data from core HP system to MS SQL Server warehouse** – **30 seconds of latency** from source to target.
- Co-existence with **Ab Initio** for data transformations.
- Microstrategy** for reporting & analysis.

High Availability for 1 Billion Debit Cards p/mth & Near Real-time Business Intelligence



"We choose GoldenGate as our infrastructure solution of choice because data integrity is so important to us."
- Joe Ramos, Director, Engineering, VISA

CONFIDENTIAL - DO NOT DISTRIBUTE

GOLDENGATE

High Availability

Case Study: Retail Decisions

24x7 Fraud Detection & Payment Processing for Blue-Chip Retailers



Business Challenges:

- Typical Service Level Agreements dictate **99.95% availability** & aggressive sub-second average response times
- Must ensure **quick, massive scalability**
- High cost of downtime -- ReD's clients lose millions of dollars per hour
- Global clients; data centers on 4 continents

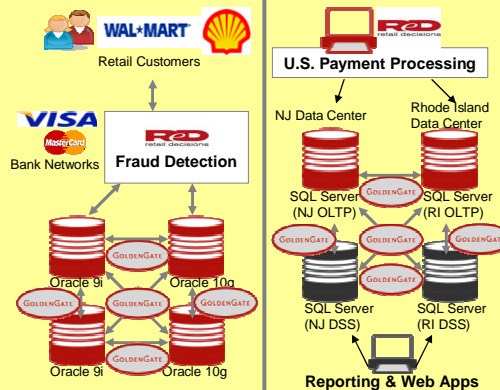
GoldenGate Active-Active Solutions:

- Fraud Detection** (ReDShield) using Oracle 9i and 10g databases
- U.S. Payment Processing** system, using SQL Server databases – also supports data access for Web apps and Reporting

Results:

- "Lightening Fast"
- Time to recover: ZERO minutes**
- Reduced database license & infrastructure costs

Active-Active for Two Application Environments



"We needed a mega-scalable architecture capable of handling increasing e-commerce traffic, while meeting our customers' stringent SLAs." - Chris Uriarte, CTO

Financial Services/Banking

CONFIDENTIAL - DO NOT DISTRIBUTE

GOLDENGATE

Transaction Offloading for Low Latency

Case Study: Sabre Holdings

Saving \$ Millions Annually with Database Tying

Sabre Holdings

Business Challenges:

- Improve system availability and performance for customers
- Reduce TCO with a "Zero Cost to Scale" to handle growth in online lookers & shoppers
- Needed low latency data between OLTP and search engine database

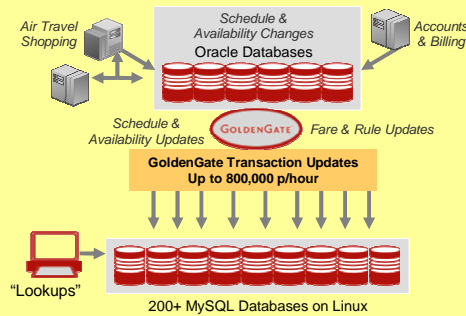
GoldenGate Solution:

- Database Tying enables high availability, top performance, and unlimited scalability
- Off-loaded query activity from costly OLTP systems ("lookers vs. bookers")
- Supported scale from 50 to 200+ lower-cost read-only targets on MySQL/Linux
- Very low impact and handles high volumes

Results:

- 80% TCO reduction – millions of dollars in savings each year
- Pushing up to 800,000 updates/hour in real time to target systems

Database Tying Enables Top Performance, Unlimited Scalability



"GoldenGate...provided the glue to move the data across systems, evolving our overall approach in parallel with the future of online travel commerce."
- James Harding, VP of Systems Planning & Performance, Sabre

Travel & Hospitality

CONFIDENTIAL - DO NOT DISTRIBUTE

GOLDENGATE

Integration with Messaging systems

Case Study: TravelCLICK

Innovating Hotelier Business with Real-Time Data Integration and EDA



Business Challenges:

- Optimize hotel inventory allocation across multiple channels including Internet, Global Distribution systems, Online travel agencies & hotel websites.
- Integrate data in real-time from online sources & provide consolidated view across hotelier business channels.
- Send hotel rates and availability data with lowest latency to subscribers.
- Hotel channels require "push" interfaces – hence JMS/XML capabilities

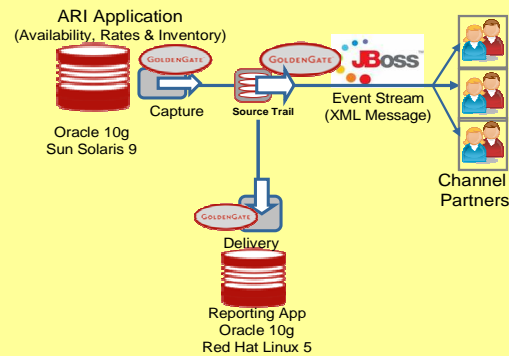
GoldenGate Solution:

- Capture transactional data from Oracle sources to provide up-to-date availability, rates and inventory (ARI) for all online booking channels
- Event Driven Architecture (EDA) support allows multiple subscriptions to the same event stream via a durable JMS topic, allowing scalability to add customers seamlessly, without increasing infrastructure cost.

Benefits:

- Reusable event steam that can drive other business processes triggered by rate and inventory changes.
- Efficient and scalable mechanism to generate events that are fundamental to event-driven SOA.

Real-Time Data Feeds to Messaging System Used by Business Partners



"GoldenGate enables us to provide real-time updates across online channels — so hoteliers receive accurate, timely information and can therefore make better decisions. We are also impressed with the extremely light footprint GoldenGate has on our production systems."
- David Marshall, Chief Architect, TravelCLICK

Travel & Hospitality

CONFIDENTIAL - DO NOT DISTRIBUTE

GOLDENGATE

Challenges

Incremental Validation

In operational BI systems the data across source and target environments doesn't undergo much transformation. With continuous integration, any data discrepancies between the operational and BI systems can have a significant impact on the real-time enterprise since the data is being updated in place, rather than replaced in batch style. Once data between sources and the BI system goes out of sync – applications that operate on the system can no longer work with stale data due to real-time expectations. The systems can diverge due to a variety of reasons. User errors, application errors, disk corruption, and infrastructure problems can lead to out-of-sync conditions. Inaccurate data may lead to decision-making based on inaccurate information, failed service level agreements, and ultimately, financial and legal risk exposures.

GoldenGate provides a software product called Veridata that performs a high speed hash compare of data sets between the source and target side tuples using multiple passes, and reports any discrepancies between the systems. Compression algorithms are used to compress the amount of data that's used in the compare stages. A key open problem is that as the data sizes in application tables continues to grow into terabytes, the amount of time taken to do the validation grows linearly because of the cost of scanning the data on both sides. When the data is out of sync – the re-instantiation of the target side is a time consuming, and laborious step. There are ways to reduce the data sets by reducing the scan sets in order to do the comparison exclusively on incremental data changes. For example, integration with bitmap metadata used by database vendors to identify changed data pages for backup purposes would help. Not enough work has been done in this area and attention from the research community is welcome.

Performance of Apply side API

With the advent of specialized appliances in the market, we find that there are high speed loading interfaces to support batch loads but not high performing APIs for trickle feeds. Many of these systems have ODBC support but with poor performance. Especially when multiple processes share the workload and do continuous loading, the system performance degrades significantly. System designers of newer data stores should recognize the importance of real-time data coming into the system, and start supporting faster native APIs to load continuous real-time data.

In traditional systems, the bottleneck with latency is on the load side because while multiple clients generate the workload on the source system, the Apply side cannot easily run with an equivalent number of parallel processes due to transaction consistency requirements to satisfy online querying. As source systems are adding multiple cores, and faster processors – log generation rates are increasing, apply side performance to support will become a serious limitation

High Availability

Oracle has recently introduced Exadata, a combination of storage software from Oracle and industry-standard hardware from HP. In a system like this, HA features are automatically inherited because of classic recovery in the database. However, addressing outages for newer data warehousing appliances, or column-oriented specialized storage appliances pose a problem in satisfying compliance requirements especially in the financial sector. Dual active requirements or logical open at a target system in an efficient manner is a problem.

Working with Teradata database kernel teams, GoldenGate has implemented scalable replication across multiple Teradata systems to support real-time requirements. Designers/implementers of newer data stores will feel pressure to support fast failover in real-time enterprises.

Databases also have support for non-logged changes for operations such as CTAS (Create Table as Select) , or direct loads that bypass the buffer cache – efficiently propagating these data changes using a log based data movement technology to the downstream system is a challenge.

Hybrid BI deployments

Supporting both real-time continuous updates, and traditional data loads by ETL or ELT technologies against the same BI system introduces new challenges like traditional batch management, data models for staging tables, MV refresh frequency. This is because real-time data acquisition techniques are replacing the Extract component in ETL/ELT work flows.

Conclusion

In light of the new emerging technology trends, advances in hardware, ubiquitous connectivity and emphasis on heterogeneity, standards and cost reduction – conventional data management frameworks need to evolve. A significant problem across real-time BI enterprises is real-time acquisition, and real-time movement of data without application disruptions or noticeable impact to the operational system providing the source data. A majority of mission critical applications use commercial relational databases for storing enterprise transactional data and this data needs to flow across a variety of systems such as data warehouses, data marts, and enterprise service buses to meet business application requirements.

The *real-time* interval where multiple systems can derive operational business intelligence value as a feedback loop to a real-time application needs to satisfy sub-second transactional flows. Key desiderata in a data management framework to address these newer trends are the ability to move data preserving transactional semantics, the performance to meet the real-time interval, and a standard manner to work across multiple heterogeneous systems. GoldenGate calls this extension to the data management framework *Transactional Data Management* (TDM), and has implemented a software platform to meet encompassing requirements thereby enabling real-time access to real-time information.

References

1. [Stone *et al*] Michael Stonebraker, Uğur Çetintemel “One Size Fits All”: An Idea Whose Time Has Come and Gone.
- 2.
3. [Gray *et. al*] J. N. Gray, P. Helland, P. O’Neil, D. Shasha – The Dangers of Replication and a Solution – ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 1996.
4. [AI_n27385199] http://findarticles.com/p/articles/mi_m0EIN/is_2007_Sept_25/ai_n27385199/
5. [Michael *et. al*] From Databases to Dataspaces: A New Abstraction for Information Management Michael Franklin, Alon Halevy, David Maier
6. [TDWI_3008] <http://www.teradata.com/t/WorkArea/DownloadAsset.aspx?id=3008>
7. [Dayal *et. al*] Data Integration Flows for Business Intelligence - **EDBT '09**: Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology Umeshwar Dayal, Malu Castellanos, Alkis Simitsis, Kevin Wilkinson
8. [InmonDW] Building the Data Warehouse by W. H. Inmon
9. [SSIS_ITB] <http://msdn.microsoft.com/en-us/library/dd537533.aspx>
10. [Microsoft_SS2008] <http://www.microsoft.com/sqlserver/2008/en/us/whats-new.aspx>
11. [Bosworth] Data Routing Rather than Databases: The Meaning of the Next Wave of the Web Revolution to Data Management
12. [Teradata_Replication] Chapter 2: Teradata Replication Solutions Architecture: <http://www.info.teradata.com/Gen-Srch/eOnLine-Prodline-ResultsAll.cfm?pl=all&title2=replication&pid=&release=&sbrn=3&startdate=&enddate=>
13. [Deiter_2002] - **Information sharing with the Oracle database**, Proceedings of the 2nd international workshop on Distributed event-based systems. Deiter Gawlick, Shailendra Mishra, Oracle Corp.
14. [Econ_2009] http://www.economist.com/sciencetechnology/tq/displaystory.cfm?STORY_ID=13725843

Additional References

GoldenGate 10.0 Reference Guide
White Paper: GoldenGate Transactional Data Management Platform June 2005

Acknowledgements

Andrei Lessiv, David Weiss, Steve Wilkes