An Oracle White Paper
August 2012

Best Practices for Real-time Data Warehousing

ORACLE

Executive Overview

Today's integration project teams face the daunting challenge that, while data volumes are exponentially growing, the need for timely and accurate business intelligence is also constantly increasing. Batches for data warehouse loads used to be scheduled daily to weekly; today's businesses demand information that is as fresh as possible. The value of this real-time business data decreases as it gets older, latency of data integration is essential for the business value of the data warehouse. At the same time the concept of "business hours" is vanishing for a global enterprise, as data warehouses are in use 24 hours a day, 365 days a year. This means that the traditional nightly batch windows are becoming harder to accommodate, and interrupting or slowing down sources is not acceptable at any time during the day. Finally, integration projects have to be completed in shorter release timeframes, while fully meeting functional, performance, and quality specifications on time and within budget. These processes must be maintainable over time, and the completed work should be reusable for further, more cohesive, integration initiatives.

Conventional "Extract, Transform, Load" (ETL) tools closely intermix data transformation rules with integration process procedures, requiring the development of both data transformations and data flow. Oracle Data Integrator (ODI) takes a different approach to integration by clearly separating the declarative rules (the "what") from the actual implementation (the "how"). With ODI, declarative rules describing mappings and transformations are defined graphically, through a drag-and-drop interface, and stored independently from the implementation. ODI automatically generates the data flow, which can be fine-tuned if required. This innovative approach for declarative design has also been applied to ODI's framework for Changed Data Capture (CDC). ODI's CDC moves only changed data to the target systems and can be integrated with Oracle GoldenGate, thereby enabling the kind of real time integration that businesses require.

This technical brief describes several techniques available in ODI to adjust data latency from scheduled batches to continuous real-time integration.

Introduction

The conventional approach to data integration involves extracting all data from the source system and then integrating the entire set—possibly using an incremental strategy—in the target system. This approach, which is suitable in most cases, can be inefficient when the integration process requires real-time data integration. In such situations, the amount of data involved makes data integration impossible in the given timeframes.

Basic solutions, such as filtering records according to a timestamp column or "changed" flag, are possible, but they might require modifications in the applications. In addition, they usually do not sufficiently ensure that all changes are taken into account.

ODI's Changed Data Capture identifies and captures data as it is being inserted, updated, or deleted from datastores, and it makes the changed data available for integration processes.

Real-Time Data Integration Use Cases

Integration teams require real-time data integration with low or no data latency for a number of use cases. While this whitepaper focuses on data warehousing, it is useful to differentiate the following areas:

- **Real-time data warehousing**
  Aggregation of analytical data in a data warehouse using continuous or near real-time loads.
- **Operational reporting and dashboards**
  Selection of operational data into a reporting database for BI tools and dashboards.
- **Query Offloading**
  Replication of high-cost or legacy OLTP servers to secondary systems to ease query load.
- **High Availability / Disaster Recovery**
  Duplication of database systems in active-active or active-passive scenarios to improve availability during outages.
- **Zero Downtime Migrations**
  Ability to synchronize data between old and new systems with potentially different technologies to allow for switch-over and switch-back without downtime.
- **Data Federation / Data Services**
  Provide virtual, canonical views of data distributed over several systems through federated queries over heterogeneous sources.

Oracle has various solutions for different real-time data integration use cases. Query offloading, high availability/disaster recovery, and zero-downtime migrations can be handled through the Oracle GoldenGate product that provides heterogeneous, non-intrusive and highly performant changed data capture, routing, and delivery. In order to provide no to low latency loads, ODI has various alternatives for real-time data warehousing through the use of CDC mechanism, including the integration with Oracle GoldenGate. This integration also provides seamless operational reporting. Data federation and data service use cases are covered by Oracle Data Service Integrator (ODSI).

Architectures for Loading Data Warehouses

Various architectures for collecting transactional data from operational sources have been used to populate data warehouses. These techniques vary mostly on the latency of data integration, from daily batches to continuous real-time integration. The capture of data from sources is either performed through incremental queries that filter based on a timestamp or flag, or through a CDC mechanism that detects any changes as it is happening. Architectures are further distinguished between pull and push operation, where a pull operation polls in fixed intervals for new data, while in a push operation data is loaded into the target once a change appears.
A daily batch mechanism is most suitable if intra-day freshness is not required for the data, such as longer-term trends or data that is only calculated once daily, for example financial close information. Batch loads might be performed in a downtime window, if the business model doesn't require 24 hour availability of the data warehouse. Different techniques such as real-time partitioning or trickle-and-flip[1] exist to minimize the impact of a load to a live data warehouse without downtime.

	Batch	**Mini-Batch**	**Micro-Batch**	**Real-Time**
Description	Data is loaded in full or incrementally using a off-peak window.	Data is loaded incrementally using intra-day loads.	Source changes are captured and accumulated to be loaded in intervals.	Source changes are captured and immediately applied to the DW.
Latency	Daily or higher	Hourly or higher	15min & higher	sub-second
Capture	Filter Query	Filter Query	CDC	CDC
Intialization	Pull	Pull	Push, then Pull	Push
Target Load	High Impact	Low Impact, load frequency is tuneable		
Source Load	High Impact	Queries at peak times necessary	Some to none depending on CDC technique	

[1] See also: Real-Time Data Warehousing: Challenges and Solutions by Justin Langseth (http://dssresources.com/papers/features/langseth/langseth02082004.html)

IMPLEMENTING CDC WITH ODI

Change Data Capture as a concept is natively embedded in ODI. It is controlled by the modular Knowledge Module concept and supports different methods of CDC. This chapter describes the details and benefits of the ODI CDC feature.

Modular Framework for Different Load Mechanisms

ODI supports each of the described data warehouse load architectures with its modular Knowledge Module architecture. Knowledge Modules enable integration designers to separate the declarative rules of data mapping from their technical implementation by selecting a best practice mechanism for data integration. Batch and Mini-Batch strategies can be defined by selecting Load Knowledge Modules (LKM) for the appropriate incremental load from the sources. Micro-Batch and Real-Time strategies use the Journalizing Knowledge Modules (JKM) to select a CDC mechanism to immediately access changes in the data sources. Mapping logic can be left unchanged for switching KM strategies, so that a change in loading patterns and latency does not require a rewrite of the integration logic.

Methods for Tracking Changes using CDC

ODI has abstracted the concept of CDC into a journalizing framework with a JKM and journalizing infrastructure at its core. By isolating the physical specifics of the capture process from the process of detected changes, it is possible to support a number of different techniques that are represented by individual JKMs:

Non-invasive CDC through Oracle GoldenGate

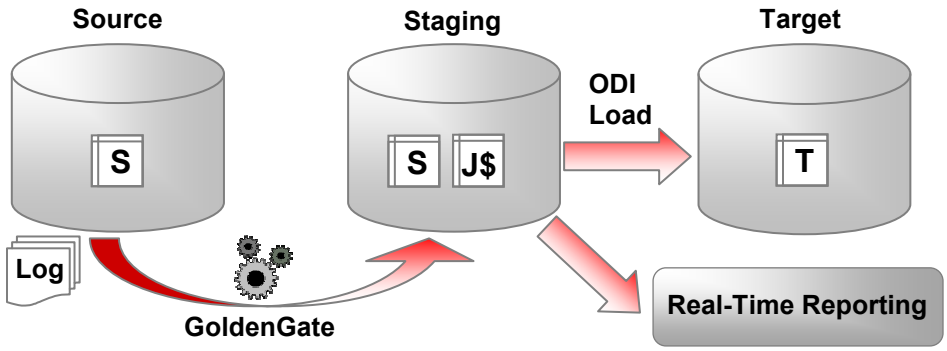

Figure 1: GoldenGate-based CDC

Oracle GoldenGate provides a CDC mechanism that can process source changes non-invasively by processing log files of completed transactions and storing these captured changes into external Trail Files independent of the database. Changes are then reliably transferred to a staging database. The JKM uses the metadata managed by ODI to generate

all Oracle GoldenGate configuration files, and processes all GoldenGate-detected changes in the staging area. These changes will be loaded into the target data warehouse using ODI's declarative transformation mappings. This architecture enables separate real-time reporting on the normalized staging area tables in addition to loading and transforming the data into the analytical data warehouse tables.

Database Log Facilities

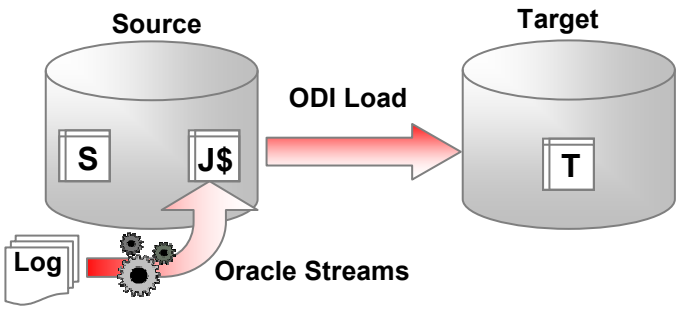

Figure 2: Streams-based CDC on Oracle

Some databases provide APIs and utilities to process table changes programmatically. For example, Oracle provides the Streams interface to process log entries and store them in separate tables. Such log-based JKMs have better scalability than trigger-based mechanisms, but still require changes to the source database. ODI also supports log-based CDC on DB2 for iSeries using its journals.

Database Triggers

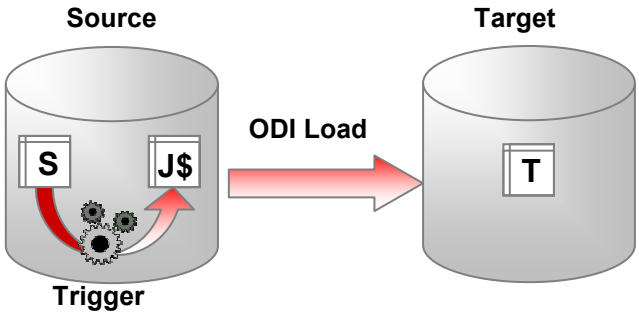

Figure 3: Trigger-based CDC

JKMs based on database triggers define procedures that are executed inside the source database when a table change occurs. Based on the wide availability of trigger mechanisms in databases, JKMs based on triggers are available for a wide range of sources such as Oracle, IBM DB2 for iSeries and UDB, Informix, Microsoft SQL Server, Sybase, and others. The disadvantage is the limited scalability and performance of trigger procedures, making them optimal for use cases with light to medium loads.

Oracle Changed Data Capture Adapters

Oracle also provides separate CDC adapters that cover legacy platforms such as DB2 for z/OS, VSAM CICS, VSAM Batch, IMS/DB and Adabas. These adapters provide performance by capturing changes directly from the database logs.

Source databases supported for ODI CDC

Database	Log-based CDC			Trigger-based CDC
	JKM Oracle GoldenGate	Database Log Facilities	Oracle CDC Adapters	
Oracle	●	●		●
MS SQL Server	●		●	●
Sybase ASE	●			●
DB2 UDB	●			●
DB2 for iSeries	●[2]	●		●
DB2 for z/OS	●[2]		●	
Informix, Hypersonic DB				●
Teradata, Enscribe, MySQL, SQL/MP, SQL/MX	●[2]			
VSAM CICS, VSAM Batch, IMS/DB, Adabas			●	

[2] Requires customization of Oracle GoldenGate configuration generated by JKM

Publish-and-Subscribe Model

The ODI journalizing framework uses a publish-and-subscribe model. This model works in three steps:

1. An identified subscriber, usually an integration process, subscribes to changes that might occur in a datastore. Multiple subscribers can subscribe to these changes.
2. The Changed Data Capture framework captures changes in the datastore and then publishes them for the subscriber.
3. The subscriber—an integration process—can process the tracked changes at any time and consume these events. Once consumed, events are no longer available for this subscriber.

ODI processes datastore changes in two ways:

- **Regularly in batches (pull mode)**—for example, processes new orders from the Web site every five minutes and loads them into the operational datastore (ODS)
- **In real time (push mode) as the changes occur—**for example, when a product is changed in the enterprise resource planning (ERP) system, immediately updates the on-line catalog

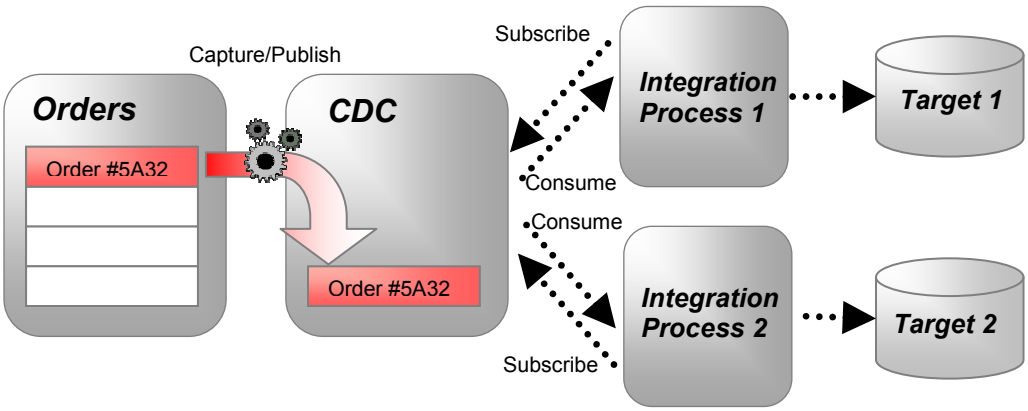

Figure 4: The ODI Journalizing Framework uses a publish-and-subscribe architecture

Processing the Changes

ODI employs a powerful declarative design approach, Extract-Load, Transform (E-LT), which separates the rules from the implementation details. Its out-of-the-box integration interfaces use and process the tracked changes.

Developers define the declarative rules for the captured changes within the integration processes in the Designer Navigator of the ODI Studio graphical user interface—without having to code. In ODI Studio, customers declaratively specify set-based maps between sources and targets, and then the system automatically generates the data flow from the set-based maps.

The technical processes required for processing the changes captured are implemented in ODI's Knowledge Modules. Knowledge Modules are scripted modules that contain database and application-specific patterns. The runtime then interprets these modules and optimizes the instructions for targets.

Ensuring Data Consistency

Changes frequently involve several datastores at one time. For example, when an order is created, updated, or deleted, it involves both the orders table and the order lines table. When processing a new order line, the new order to which this line is related must be taken into account.

ODI provides a mode of tracking changes, called Consistent Set Changed Data Capture, for this purpose. This mode allows you to process sets of changes that guarantee data consistency.

Best Practices using ODI for Real-Time Data Warehousing

As with other approaches there is no one-size-fits-all approach when it comes to Real-Time Data Warehousing. Much depends on the latency requirements, overall data volume as well as the daily change volume, load patterns on sources and targets, as well as structure and query requirements of the data warehouse. As covered in this paper, ODI supports all approaches of loading a data warehouse.

In practice there is one approach that satisfies the majority of real-time data warehousing use cases: The micro-batch approach using GoldenGate-based CDC with ODI. In this approach, one or more tables from operational databases are used as sources and are replicated by GoldenGate into a staging area database. This staging area provides a real-time copy of the transactional data for real-time reporting using BI tools and dashboards. The operational sources are not additionally stressed as GoldenGate capture is non-invasive and performant, and the separate staging area handles operational BI queries without adding load to the transactional system. ODI performs a load of the changed records to the real-time data warehouse in frequent periods of 15 minutes or more. This pattern has demonstrated the best combination of providing fresh, actionable data to the data warehouse without introducing inconsistencies in aggregates calculated in the data warehouse.

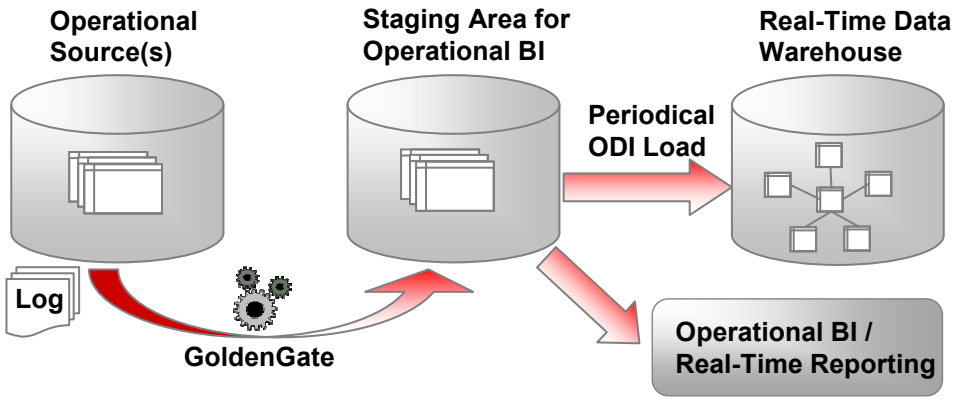

Figure 5: Micro-Batch Architecture using ODI and GoldenGate

Customer Use Case: Overstock.com

Overstock.com is using both GoldenGate and ODI to load customer transaction data into a real-time data warehouse. GoldenGate is used to capture changes from its retail site, while ODI is used for complex transformations in its data warehouse.

Overstock.com has seen the benefits of leveraging customer data in real time. When the company sends out an e-mail campaign, rather than waiting one, two, or three days, it immediately sees whether consumers are clicking in the right place, if the e-mail is driving consumers to the site, and if those customers are making purchases. By accessing the data in real time using Oracle GoldenGate and transforming it using Oracle Data Integrator, Overstock.com can immediately track customer behavior, profitability of campaigns, and ROI. The retailer can now analyze customer behavior and purchase history to target marketing campaigns and service in real time and provide better service to its customers.

Conclusion

Integrating data and applications throughout the enterprise, and presenting a consolidated view of them, is a complex proposition. Not only are there broad disparities in data structures and application functionality, but there are also fundamental differences in integration architectures. Some integration needs are data oriented, especially those involving large data volumes. Other integration projects lend themselves to an event-oriented architecture for asynchronous or synchronous integration.

Changes tracked by Changed Data Capture constitute data events. The ability to track these events and process them regularly in batches or in real time is key to the success of an event-driven integration architecture. Oracle Data Integrator provides rapid implementation and maintenance for all types of integration projects.

Best Practices for Real-time Data Warehousing
August 2012
Author: Alex Kotopoulis

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com

Oracle is committed to developing practices and products that help protect the environment

0510