Oracle Fusion SaaS Applications High Frequency Data Extraction and Ingestion on OCI

Design Patterns

October 2023

Matthieu Lombard, Oracle A-Team Cloud Solution Architects

Contents

Disclaimer

Introduction

In today's data-driven world, having access to fresh data is a fundamental aspect of staying competitive and agile. It empowers companies to make informed decisions, minimize risks, enhance operational efficiency, and meet the evolving needs of customers and markets.

This document is an attempt to describe a technical architectural pattern for Oracle Fusion Application Cloud data extract at a higher frequency than it is typically used in a Data Management and Analytics context to enable organization to meet the data recentness requirements in a Finance, Supply Chain Management or Customer Experience context.

It will first describe the various business cases and benefits for the functional domains cited above.

Then it will describe the technical architecture on Oracle Cloud Infrastructure (OCI) of the proposed solution, specifically the Data Extraction layer and the Data Transformation and Load layer.

Having close to near real-time data updates in Enterprise Resource Planning (ERP), Supply Chain Management (SCM), and Customer Experience (CX) offers significant business benefits by providing decision-makers with up-to-date information. In today's fast-paced business environment, real-time data enables organizations to make agile and informed decisions. For ERP systems, it means that financial and operational data is constantly refreshed, allowing for precise financial forecasting, inventory management, and production planning. This, in turn, helps in optimizing resource allocation, reducing costs, and ensuring that business processes are aligned with current market conditions. In SCM, real-time data enables enhanced visibility into the supply chain, allowing companies to respond promptly to disruptions, optimize inventory levels, and meet customer demand efficiently. This results in reduced lead times, lower carrying costs, and improved customer satisfaction.

In the realm of Customer Experience (CX), real-time data empowers businesses to deliver a superior level of service. When customer interactions and feedback are promptly analyzed, companies can make adjustments to improve their products, services, and customer support in real-time. It also enables personalized marketing and engagement strategies, as businesses can respond to customer behavior and preferences instantly. Consequently, CX benefits from higher customer satisfaction, loyalty, and increased revenue. Overall, having close to near real-time data updates in ERP, SCM, and CX systems is a strategic advantage that enables businesses to stay competitive and responsive in a dynamic market, fostering growth and customer-centricity.

Business Cases

Finance – Enterprise Resource Planning Use Cases

Having Up-to-date and close to "Near Real-time" ERP data from month-end book closes is fundamental for ensuring the integrity of financial reporting, facilitating informed decision-making, meeting regulatory requirements, and managing various aspects of a business effectively. It forms the foundation for sound financial management and helps a company operate efficiently and transparently.

This is critical for several reasons, such as:

- **Accurate Financial Reporting**: Up-to-date financial data ensures that a company's financial statements accurately represent its current financial position. This is crucial for shareholders, investors, lenders, and other stakeholders who rely on these reports to make informed decisions.
- **Timely Decision-Making**: Business decisions are often based on financial data. With accurate and current financial information, management can make informed decisions about budgeting, resource allocation, investments, and strategic planning in a timely manner.
- **Compliance**: Many regulatory requirements and accounting standards dictate that companies must provide timely and accurate financial reports. Failing to do so can result in legal and regulatory penalties.
- **Risk Management**: Keeping financial data current helps a company identify financial risks early. It allows for the assessment of liquidity, solvency, and profitability, helping a company take measures to mitigate potential financial issues.
- **Investor and Stakeholder Confidence**: Regular, up-to-date financial reporting can instill confidence in investors, lenders, and other stakeholders. When these parties have access to current financial data, they are more likely to trust the company and may be willing to provide capital or support.
- **Performance Monitoring**: By having current financial data, a company can monitor its performance against budgets and forecasts. This allows for quick adjustments if performance deviates from expectations.
- **Cash Flow Management**: Timely financial data helps with managing cash flow effectively. It provides insight into receivables, payables, and other factors affecting liquidity, which is crucial for day-to-day operations.
- **Tax Planning**: Current financial data is essential for tax planning. It helps in calculating and setting aside the appropriate amount of income tax and ensures compliance with tax regulations.
- **Audit Preparation**: If a company is subject to financial audits, having up-to-date financial data streamlines the audit process. Auditors can work more efficiently, reducing the burden on the company.
- **Strategic Planning**: Accurate, current financial data is vital for long-term strategic planning. It helps identify opportunities and threats, guides investment decisions, and supports the development of a clear financial roadmap.
- **Quality Control**: Regular financial closings ensure the accuracy of accounting records and help identify errors or discrepancies that need to be addressed.

Supply Chain Management Use Cases

Maintaining a near real-time data position in Supply Chain Management (SCM) is vital for the enhancement of SCM operations, informed decision-making, cost reduction, improved customer service, and risk management. This capability would empower organizations to operate with greater efficiency and competitiveness within the dynamic and fast-paced business landscape.

This is critical for several reasons:

- **Increased Visibility**: finance data provides immediate visibility into the supply chain. This visibility helps companies track the movement of goods, monitor inventory levels, and identify any disruptions or delays as they occur. This information is crucial for addressing issues promptly.
- **Enhanced Decision-Making**: Near Real-time data allows for more informed and timely decision-making. When supply chain managers have access to current information, they can make adjustments to optimize routes, reorder inventory, or reroute shipments to avoid bottlenecks or delays.
- **Improved Responsiveness**: With Near Real-time data, companies can respond rapidly to changing market conditions, customer demands, and unexpected events (such as natural disasters or disruptions in the supply chain). This agility can help a company stay competitive and meet customer expectations.
- **Better Inventory Management**: Near Real-time data helps in managing inventory efficiently. It enables companies to avoid overstocking or understocking, reducing carrying costs and minimizing the risk of stockouts.
- **Reduction of Bullwhip Effect**: The bullwhip effect is a phenomenon where small fluctuations in demand at the retail level can lead to significant demand variability further up the supply chain. Near Real-time data can help mitigate this effect by providing more accurate demand information, allowing for better planning and forecasting.
- **Cost Savings**: Near Real-time data can lead to cost savings by optimizing transportation routes, reducing lead times, and minimizing the need for emergency expedited shipments. It also helps in identifying cost-saving opportunities, such as consolidating orders**.**
- **Improved Customer Service**: Near Real-time data allows companies to provide more accurate delivery estimates to customers. This can enhance customer satisfaction and loyalty by ensuring timely deliveries and reducing the number of delivery exceptions.
- **Risk Management**: Near Real-time data helps identify and mitigate risks in the supply chain, whether related to disruptions, quality issues, or compliance concerns. Companies can take immediate action to address these risks and reduce their impact.
- **Supplier Performance Monitoring**: Near Real-time data enables companies to monitor supplier performance in real time. This ensures that suppliers are meeting agreed-upon service levels and quality standards, and deviations can be addressed promptly.
- **Quality Control**: Near Real-time data can be used to monitor the quality and condition of goods in transit. If there are issues with product quality or safety, this information can trigger immediate recalls or corrective actions.
- **Regulatory Compliance**: In industries with strict regulatory requirements, Near Real-time data can help companies ensure compliance with regulations related to product safety, traceability, and reporting.

- **Competitive Advantage**: Companies with Near Real-time supply chain data often have a competitive edge. They can react quickly to market changes, meet customer demands more effectively, and adapt to shifting market conditions.

Customer Experience Use Cases

Utilizing real-time Customer Experience data empowers businesses to promptly address customer requirements and enhance overall satisfaction. The significance of real-time customer experience data becomes evident through several key reasons, including:

- **Immediate Issue Resolution**: Real-time data enables companies to address customer issues and concerns as they occur. This can lead to quicker problem resolution, enhancing the customer's experience and preventing negative feedback or escalation.
- **Personalization**: Real-time data allows businesses to personalize interactions with customers. For example, it enables personalized product recommendations or targeted offers based on a customer's current behavior and preferences.
- **Enhanced Customer Support**: Customer support teams can use real-time data to provide more effective assistance. They have access to the customer's history, context, and recent interactions, which helps resolve issues efficiently.
- **Feedback Loop**: Real-time data provides immediate feedback on products, services, or customer interactions. This feedback can be used to make adjustments and improvements promptly, leading to better customer satisfaction.
- **Improved Decision-Making**: Business decisions related to customer experience can be made based on real-time data. This includes changes in pricing, product offerings, or service enhancements that directly impact customers.
- **Preventing Churn**: Real-time customer experience data can help identify signs of customer dissatisfaction or potential churn. Companies can take proactive measures to retain customers, such as offering incentives or resolving issues quickly.
- **Competitive Advantage**: Businesses that respond in real time to customer needs gain a competitive edge. Customers appreciate quick responses and are more likely to remain loyal to brands that provide them.
- **Market Agility**: Real-time customer data allows businesses to adapt to changing market conditions and customer expectations promptly. This agility is essential in today's fast-paced business environment.
- **Data-Driven Insights**: Real-time data provides immediate insights into customer behavior and preferences. This data is invaluable for marketing and product development strategies.
- **Increased Sales**: With real-time data, businesses can identify cross-selling and upselling opportunities in real time. This can lead to increased sales and revenue.
- **Positive Word-of-Mouth**: Quick issue resolution and personalized experiences contribute to positive word-of-mouth marketing. Satisfied customers are more likely to recommend a brand to others.
- **Brand Loyalty**: Real-time customer experience data helps build and strengthen customer loyalty. When customers feel heard and valued, they are more likely to remain loyal to a brand.
- **Reduced Costs**: Timely issue resolution and proactive customer retention efforts can reduce the costs associated with handling customer complaints, refunds, and acquisition of new customers.

Technical Solution Overview

This reference architecture is a standardized template or blueprint that provides a recommended structure and set of best practices for designing and implementing the solution to solves the business cases depicted in the previous section. It is intended to help organizations save time, reduce costs, and improve the quality, interoperability, and security of their Data Management and Analytics implementations on OCI sourcing Oracle Fusion SaaS Applications.

The proposed architectural pattern to extract the most efficiently and rapidly data out of Oracle Fusion SaaS Applications into OCI is comprised of five different elements:

1. **Orchestration and Data Movement**: Oracle Data Integrator is used to schedule BI Cloud Connector (BICC) extracts and well as the load into Autonomous database tables.
2. **Data Extraction**: BICC data extracts using Fine Grain Public View Objects.
3. **Data Extracts Storage**: Oracle Object Storage is the external storage of choice for its performance and reliability.
4. **Data Transformation and Load**: Oracle Data Integrator and Autonomous Database are composing the data transformation and loading layer with table partition and partition exchange approach.
5. **Analytics** : Oracle Analytics Cloud Oracle Analytics Cloud is a scalable and secure Oracle Cloud service that provides a full set of capabilities to explore and perform collaborative analytics for you, your workgroup, and your enterprise.

The diagram below presents the comprehensive blueprint that offers guidance, standards, and best practices for designing and deploying Oracle Fusion SaaS Applications High Frequency Data Extraction and Ingestion on OCI solutions.

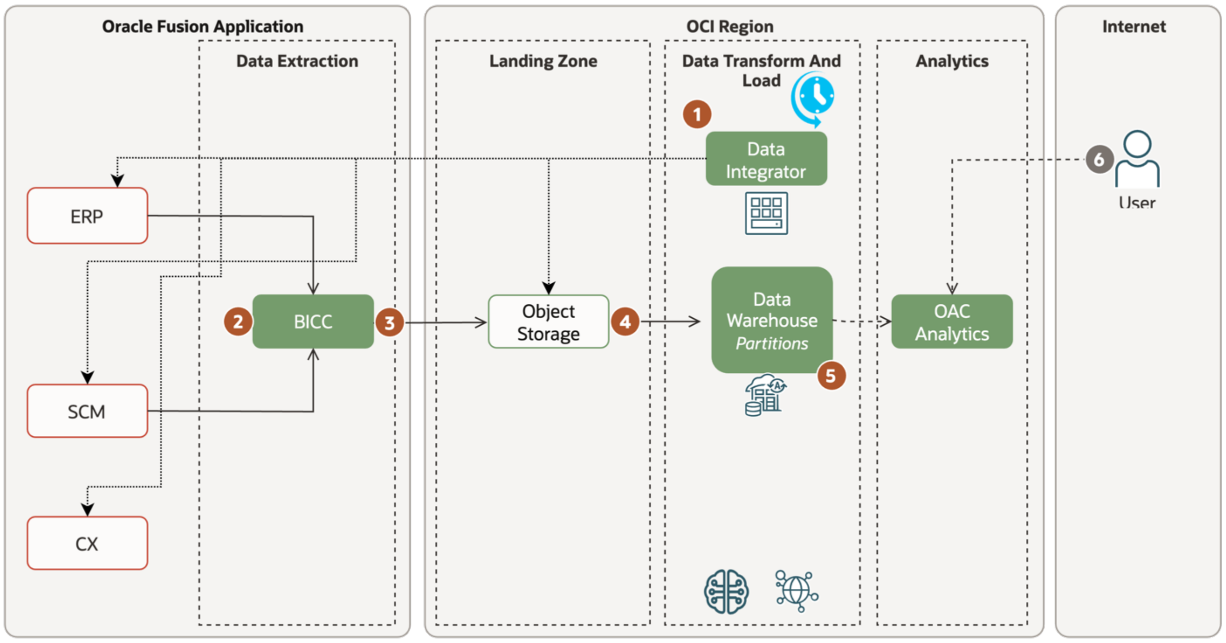
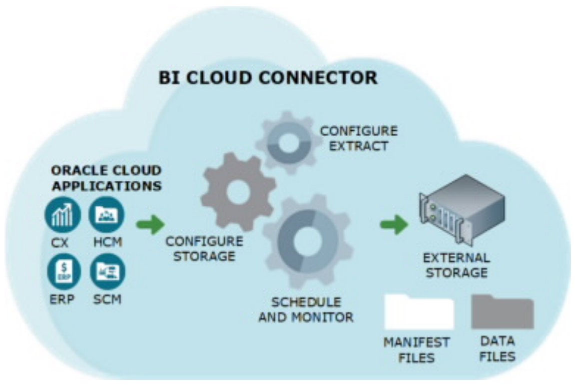


Figure 1 : Oracle Fusion SaaS Applications High Frequency Data Extraction and Ingestion on OCI Architecture

Solution Details

Extraction - BI Cloud Connector (BICC) Guidelines



The reference architecture uses Oracle Business Intelligence Cloud Connector (BICC) to extract business intelligence and other data in bulk and load it into designated external storage areas using Public View Objects (PVOs).

Requirements Gathering

Collecting data element requirements (BICC PVOs and their columns) and creating a solid data model are foundational steps in the design of a data warehouse or data lake house. These activities promote data understanding, quality, integration, and accessibility, while also supporting data governance, security, scalability, and efficient data management and analysis. A thoughtful and well-structured approach to data modeling and requirements gathering is essential for the success of data-related projects.

To reduce the BICC extraction time of PVOs, it is critical to gather accurate data requirements to identify the minimum BICC PVOs column sets to only extract the columns that are required by the business users. To do so, the recommendation is to thoroughly review BICC Lineage documents to align the data extraction with the business data elements needs:

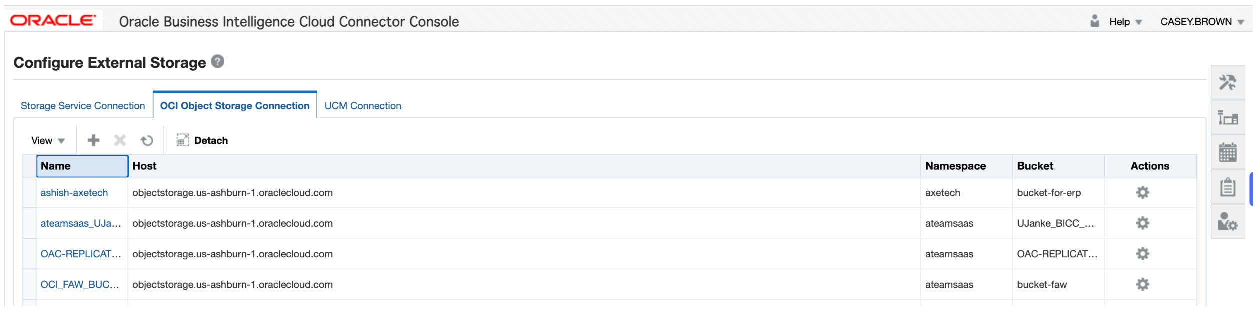
- ERP / SCM
  - Data Lineage Spreadsheet with ERP Fusion Database column to PVO column mapping
  - Oracle Fusion Cloud SCM Extract Data Stores for SCM
  - Oracle Fusion Cloud Procurement Extract Data Stores for Procurement
  - Oracle Fusion Cloud Financials Extract Data Stores for Financials
- CX
  - Data Lineage Spreadsheet with CX Fusion Database column to PVO column mapping
  - Oracle Fusion Cloud Customer Experience Extract Data Stores for CX Sales and Fusion Service

Operational Guidelines

Oracle Object Storage and Universal Content Management (UCM) storage are two different storage solutions, each with its own set of advantages and use cases. When extracting data out of Oracle Fusion Cloud applications using BI Cloud Connector, the choice between the two storage options depends on specific requirements and objectives. Here are some benefits of Oracle Object Storage over UCM storage for this particular scenario:

1. **Cost-Effectiveness:** Oracle Object Storage typically offers more cost-effective storage solutions when compared to Universal Content Management, which may have additional features and capabilities that you may not need for data extraction and storage.
2. **Scalability:** Oracle Object Storage is highly scalable and can handle large volumes of data, making it well-suited for data extraction from Oracle Fusion Cloud applications. You can increase storage capacity as needed without major disruptions.
3. **Integration:** Oracle Object Storage is tightly integrated with Oracle Cloud services, making it a seamless choice when working with other Oracle Cloud services, such as BI Cloud Connector. The integration allows for efficient data transfer and management.
4. **Security:** Oracle Object Storage provides robust security features, including encryption, access controls, and user authentication, which are important when storing and extracting sensitive business data.
5. **Data Retention:** Object Storage can be configured for data retention and compliance requirements, ensuring that extracted data is securely stored and maintained according to regulatory standards.
6. **Reliability:** Oracle Object Storage is designed for high availability and redundancy. Your data is distributed across multiple data centers, reducing the risk of data loss or downtime.
7. **Simplified Management:** Object Storage's user-friendly interface and API support streamline data management and facilitate easy integration with BI Cloud Connector, simplifying the data extraction process.
8. **Standardized Access:** Oracle Object Storage follows industry-standard protocols, such as RESTful APIs and HTTP, making it accessible and compatible with a wide range of tools and applications, including BI Cloud Connector.
9. **Data Lifecyle Management:** Object Storage supports data lifecycle management policies, allowing you to automate the movement and retention of data, which can be especially valuable for data extraction processes.

The recommendation here is to set up Object Storage Cloud destination for BI Cloud Connector, as described in this blog.


ORACLE
Oracle Business Intelligence Cloud Connector Console
Help
CASEY.BROWN
Configure External Storage
Storage Service Connection
OCI Object Storage Connection
UCM Connection
View
Detach

Name	Host	Namespace	Bucket	Actions
ashish-axetech	objectstorage.us-ashburn-1.oraclecloud.com	axetech	bucket-for-erp	⚙
ateamsaas_UJa...	objectstorage.us-ashburn-1.oraclecloud.com	ateamsaas	UJanke_BICC_...	⚙
OAC-REPLICAT...	objectstorage.us-ashburn-1.oraclecloud.com	ateamsaas	OAC-REPLICAT...	⚙
OCI_FAW_BUC...	objectstorage.us-ashburn-1.oraclecloud.com	ateamsaas	bucket-faw	⚙

Figure 2: BICC Object Storage Destination Configuration

Public View Objects Considerations

Use Fine Grain PVOs

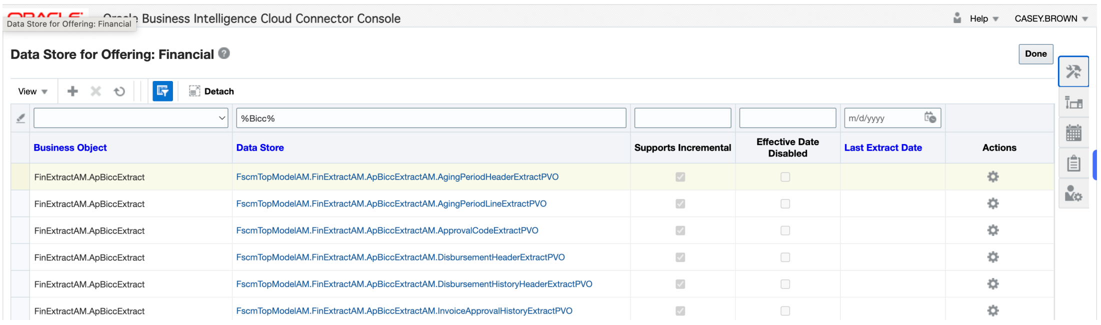
Public View Object (PVO) typically refers to a component used in the BICC platform for creating, managing, and displaying views of data. These views can be in the form of tables, charts, or reports and are designed for use by business users and analysts for data analysis and reporting purposes. Public view objects are a key part of the Oracle BI Cloud Connector, which is used to connect and integrate data from various sources into the Oracle BI platform. Here are some key characteristics and purposes of public view objects:

1. **Data Presentation:** Public view objects are used to present data to business users in a user-friendly and meaningful way. They can transform raw data into visually appealing tables, charts, and reports.
2. **Business Layer:** Public view objects are part of the business layer in the Oracle BI platform. They sit on top of the physical data sources and provide a logical and organized view of the data.
3. **Security and Permissions:** Public view objects can be associated with security and permissions, ensuring that users only access the data they are authorized to view.
4. **Customization:** Users can customize public view objects to suit their specific reporting and analysis needs. This includes choosing the visualization type, formatting, and layout.
5. **Data Filters:** Public view objects can include predefined data filters, allowing users to interactively filter and drill down into the data.
6. **Ease of Use:** The goal of public view objects is to make data analysis and reporting as user-friendly and intuitive as possible, even for users without technical or SQL query-writing skills.

When extracting data from Fusion SaaS, there are usually two types of PVOs one can use:

1. "Regular" PVOs : these PVOs are the original typical PVOs that correspond to a join between a base Fusion SaaS database table and other peripheral Fusion SaaS database tables. Executing these PVOs can in some case result in longer extracting time as the database
2. "Fine Grain" PVOs: these PVOs are PVOs that have been optimized for Data Extracts that correspond to a single Fusion SaaS database table. Below are some Fine Grain PVOs. Note that they have ***BiccExtract*** in their name.
   a. FscmTopModelAM.FinExtractAM.Gl***BiccExtract***AM.LedgerExtractPVO
   b. FscmTopModelAM.FinExtractAM.Gl***BiccExtract***AM.JournalLineExtractPVO
   c. FscmTopModelAM.FinExtractAM.Xla***BiccExtract***AM.SubledgerJournalTransactionEntityExtractPVO
   d. …

The recommendation here is to use Fine Grain PVOs when extracting data from Fusion SaaS using BICCC.


Data Store for Offering: Financial
Business Intelligence Cloud Connector Console
Help
CASEY.BROWN
Done
View
Detach
%Bicc%
m/d/yyyy

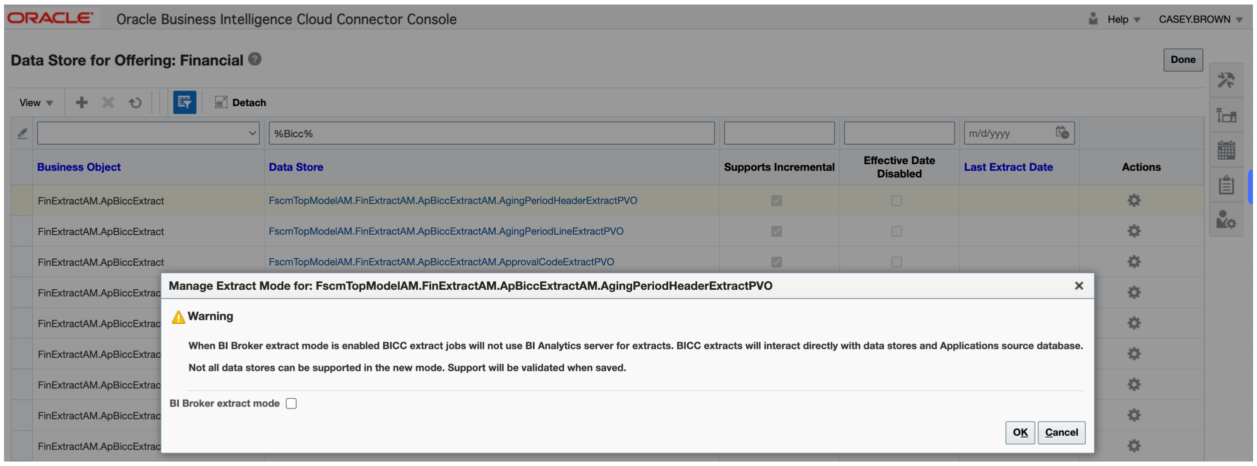
Data Store for Offering: Financial

Business Object	Data Store	Supports Incremental	Effective Date Disabled	Last Extract Date	Actions
FinExtractAM.ApBiccExtract	FscmTopModelAM.FinExtractAM.ApBiccExtractAM.AgingPeriodHeaderExtractPVO	✓	☐		⚙
FinExtractAM.ApBiccExtract	FscmTopModelAM.FinExtractAM.ApBiccExtractAM.AgingPeriodLineExtractPVO	✓	☐		⚙
FinExtractAM.ApBiccExtract	FscmTopModelAM.FinExtractAM.ApBiccExtractAM.ApprovalCodeExtractPVO	✓	☐		⚙
FinExtractAM.ApBiccExtract	FscmTopModelAM.FinExtractAM.ApBiccExtractAM.DisbursementHeaderExtractPVO	✓	☐		⚙
FinExtractAM.ApBiccExtract	FscmTopModelAM.FinExtractAM.ApBiccExtractAM.DisbursementHistoryHeaderExtractPVO	✓	☐		⚙
FinExtractAM.ApBiccExtract	FscmTopModelAM.FinExtractAM.ApBiccExtractAM.InvoiceApprovalHistoryExtractPVO	✓	☐		⚙

Figure 3: BICC Fine Grain PVOs

Enable BI Broker Mode

When BI Broker extract mode is enabled, extract jobs don't use the BI Server platform component. They interact directly with data stores and the Cloud Applications source database. To enable it, select BI Broker extract mode in the Manage Extract Mode dialog and click OK. A check is then performed to verify that BI Broker mode is supported for the data store, and if not an error is displayed.



Data Store for Offering: Financial

	Business Object	Data Store	Supports Incremental	Effective Date Disabled	Last Extract Date	Actions
		%Bicc%			m/d/yyyy	
	FinExtractAM.ApBiccExtract	FscmTopModelAM.FinExtractAM.ApBiccExtractAM.AgingPeriodHeaderExtractPVO	✓	☐		
	FinExtractAM.ApBiccExtract	FscmTopModelAM.FinExtractAM.ApBiccExtractAM.AgingPeriodLineExtractPVO	✓	☐		
	FinExtractAM.ApBiccExtract	FscmTopModelAM.FinExtractAM.ApBiccExtractAM.ApprovalCodeExtractPVO	✓	☐		
	FinExtractAM.ApBiccExtrac					
	FinExtractAM.ApBiccExtrac					
	FinExtractAM.ApBiccExtrac					
	FinExtractAM.ApBiccExtrac					
	FinExtractAM.ApBiccExtrac					
	FinExtractAM.ApBiccExtrac					

Manage Extract Mode for: FscmTopModelAM.FinExtractAM.ApBiccExtractAM.AgingPeriodHeaderExtractPVO

Warning

When BI Broker extract mode is enabled BICC extract jobs will not use BI Analytics server for extracts. BICC extracts will interact directly with data stores and Applications source database.

Not all data stores can be supported in the new mode. Support will be validated when saved.

BI Broker extract mode ☐

Figure 4: BICC Object Storage Destination Configuration

Scheduling Considerations

BICC number of concurrent threads applies at BICC node level and limits parallel PVO extracts running on a single BICC node across all JOBs. OBIEE does not have any limits for concurrent requests (logical SQLs.

To better understand BICC and OBIEE load balancing consider the following example:

- You created a single JOB with 10 VO extracts and the default BICC threads set to 5.
- BICC will start the JOB on a single node, spawning first 5 VO extracts.
- OBIEE will distribute the 5 VO extracts between its two cluster nodes, 3 VOs going to node1 and 2 VOs to node2.
- BICC will maintain the maximum concurrency = 5, spawning more extracts as soon as OBIEE complete any of its jobs on node1 or node2.
- BICC will keep spawning more requests, maintaining the concurrency = 5, until it finishes all extracts.

Some recommended design practices for BICC job definition are :

- Decouple heavy VO extracts into separate jobs. Having them included into common jobs could result in them running late in the cycle and extending the extract window.
- Use Group Number and Group Item Priority values to manage the order of VO executions within a single JOB
- Use Group Item Priority to set the order of VO extracts within a group with the same 'Group Number' in a single job. BICC would prioritize the jobs executions starting with lower Group Item Priority and move up through the list
- If you configure both Data and Primary Keys (PKs) extracts, then create two separate jobs, one for data extract and the other one for PKs.

The practical scheduling, smart jobs design, use of group numbers and group item priorities, decoupling data from PKs into separate jobs should help you to ensure most efficient extracts execution as well as load balancing in BICC and OBIEE clustered environment.

Externalize BICC Last Extract Date

Managing the "last extract date" in Oracle BI Cloud Connector (BICC) is important for maintaining data freshness and ensuring that you are extracting the most up-to-date data from your source systems. This date typically represents the timestamp of the last successful data extraction. Here are steps to manage the last extract date in BICC:

1. **Define Data Sources:** In BICC, you first need to define the data sources you want to extract data from. This involves configuring the connections to your source systems, which could be databases, cloud applications, or other data repositories.
2. **Data Extract Configuration:** Within BICC, you'll set up data extract configurations. This includes specifying the tables, views, or data objects you want to extract data from.
3. **Incremental Extraction:** To manage the last extract date, you should configure incremental extraction. This means that during each extraction, BICC will only extract data that has been added or modified since the last successful extraction.
4. **Last Extract Date Field:** In your source system, ensure that there is a field or column that captures the timestamp of when a record was added or modified. This is the field that BICC will use to track changes and perform incremental extractions.
5. **Initial Extraction:** When setting up BICC for the first time, you'll typically perform an initial full extraction of data. This will load all existing data from the source system into your target data store (e.g., Oracle Autonomous Data Warehouse, Oracle Database, etc.).
6. **Save Last Extract Date:** After each successful extraction, BICC will record the timestamp of that extraction as the "last extract date." This date is saved internally in BICC for reference in subsequent extractions.
7. **Scheduled Extracts:** BICC allows you to schedule data extraction jobs. You can set up a regular schedule (e.g., daily, weekly) to automatically run data extraction jobs. BICC will use the "last extract date" to determine which data to extract during each run.
8. **Monitoring and Alerts:** Implement monitoring and alerts to be notified in case of any extraction failures. You should be aware of any interruptions in data extraction to ensure that your data remains up to date.
9. **Historical Data:** If your source system does not have a timestamp or modified date for historical data, you may need to consider other methods for capturing historical data, such as using a separate historical snapshot table.
10. **Data Validation:** After each extraction, it's essential to validate the data to ensure that the incremental extraction process is working correctly and that the data is consistent.

By following these steps and configuring BICC to perform incremental extractions based on the last extract date, you can ensure that your data remains fresh and that you are not unnecessarily re-extracting data that hasn't changed since the last extraction. This approach is efficient and ensures that you're working with the most up-to-date data for your reporting and analytics.

To allow better control and resiliency on the BICC VO incremental extract, the recommendation is to

- Maintain Last Extract Date in a driver table in Autonomous Database
- Run BICC Extracts from Oracle Data Integrator as External Scheduler. This requires an update to the LKM BICC to ADW External Table to set the date filter in the extract job step
- Pass the external last extract date to BICC SOAP / REST call. This requires on the fly BICC PVO customization using BICC REST API to set the date filter as described here.
- Use use the last extract date in the ODI LKM options
- Update driver table after successful run with the Last Extract Date as well as the ODI job details (Session Name, start and end, number of records…)
- To minimize failure, implement retries and resiliency approaches described in the Best Practices for Building Resilient Integrations with Oracle Data Integrator here.

Transform and Load

Autonomous Database Design

Using partitioned tables when loading data into Oracle Autonomous Database offers several benefits, especially when dealing with large volumes of data and data warehousing / data lake house scenarios. Partitioning is a database design technique that divides large tables into smaller, more manageable pieces called partitions.

Each partition can be stored separately and has its own set of data. Here are the benefits of using partitioned tables in Oracle Autonomous Database for data loading and management:

Benefits	Details
Improved Data Load Performance	Loading data into a partitioned table can be significantly faster than loading into a non-partitioned table. The database can leverage parallelism and load data into individual partitions concurrently, reducing the overall load time
Reduced Maintenance Overhead	Partitioning allows for efficient data management by isolating data into separate partitions. This can simplify data maintenance tasks such as data purging, archiving, and backup and restore operations
Faster Data Retrieval	When querying a partitioned table, the database can skip reading partitions that do not contain relevant data. This results in improved query performance, especially when using appropriate partition pruning techniques.
Enhanced Query Performance	Partitioning can improve the performance of queries that involve date range filtering or other criteria that align with partition keys. Queries can be highly selective, scanning only relevant partitions.
Optimized Backup and Restore	Partitioned tables allow for efficient backup and restore operations. You can selectively back up and restore individual partitions, reducing the time and storage space required.
Easier Data Archiving	When older data needs to be archived or removed, partitioning makes it easier to identify and manage the specific partitions that need to be archived or purged
Reduced Indexing Overhead	With partitioned tables, you can create local indexes on each partition, which are smaller and more efficient than global indexes. This reduces the overhead associated with indexing large tables.
Simplified Data Management	When working with large datasets, partitioned tables simplify data management by providing a clear structure for organizing and accessing data. Data is logically grouped based on partition keys
Consistent Performance	By isolating data into partitions, performance of the system remains consistent even as the volume of data grows. Partition pruning techniques help maintain query performance.
Scalability	Partitioned tables support horizontal partitioning, allowing you to add new partitions as data grows, thereby facilitating scalability

In summary, using partitioned tables in Oracle Autonomous Database provides a range of benefits, including improved data load performance, faster data retrieval, simplified data management, efficient data maintenance, and enhanced query performance. When working with large datasets, partitioning is a valuable technique for optimizing database operations and improving the overall performance and manageability of the database.

The Oracle Autonomous Database provide a feature named Automatic Partitioning. It analyzes and automates partition creation for tables and indexes of a specified schema to improve performance and manageability in Autonomous Database.

Automatic partitioning, when applied, is transparent and does not require any user interaction or maintenance.

Note: Automatic partitioning does not interfere with existing partitioning strategies and is complementary to manual partitioning in Autonomous Database. Manually partitioned tables are excluded as candidates for automatic partitioning.

Oracle Data Integrator Guidelines

Prerequisites in Autonomous Database

There are a series of prerequisites to enable in the Autonomous Database to allow ODI to leverage partition loading. These prerequisites are described below with a sample query based on the extract of *FscmTopModelAM_FinExtractAM_GlBiccExtractAM_JournalLineExtractPVO* view object and load into *GL_DW_JOURNALLINES* table.

Steps	Sample Query
Investigate data distribution in GL_DW_JOURNALLINES to identify partition key	`SELECT DISTINCT GLJELINESPERIODNAME FROM "GL_DW_JOURNALLINES";`
Create a copy of GL_DW_JOURNALLINES	`CREATE TABLE " _DW_JOURNALLINES_INIT" as SELECT * FROM "GL_DW_JOURNALLINES" ;`
Drop ATGL_DW_JOURNALLINES	`drop table "GL_DW_JOURNALLINES" cascade constraints PURGE;`
Create temporary table GL_DW_JOURNALLINES_TEMP for incremental load	CREATE TABLE "GL_DW_JOURNALLINES_TEMP" ("JEHEADERID" NUMBER(18,0), "JELINENUM" NUMBER(18,0), "GLJELINESLASTUPDATEDATE" TIMESTAMP (6), "GLJELINESPERIODNAME" VARCHAR2(15) COLLATE "USING_NLS_COMP", "GLJELINESDESCRIPTION" VARCHAR2(240) COLLATE "USING_NLS_COMP", "GLJELINESLEDGERID" NUMBER(18,0), "GLJELINESCURRENCYCODE" VARCHAR2(15) COLLATE "USING_NLS_COMP", "GLJELINESCODECOMBINATIONID" NUMBER(18,0), "GLJELINESSTATUS" VARCHAR2(1) COLLATE "USING_NLS_COMP", "GLJELINESENTEREDDR" NUMBER, "GLJELINESENTEREDCR" NUMBER, "GLJELINESACCOUNTEDDR" NUMBER, "GLJELINESACCOUNTEDCR" NUMBER, "GLJELINESSTATAMOUNT" NUMBER, "GLJELINESLINETYPECODE" VARCHAR2(20) COLLATE "USING_NLS_COMP", "GLJELINESATTRIBUTECATEGORY" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESATTRIBUTE1" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESATTRIBUTE2" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESATTRIBUTE3" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESATTRIBUTE4" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESATTRIBUTE5" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESATTRIBUTE6" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESATTRIBUTE7" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESATTRIBUTE8" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESATTRIBUTE9" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESATTRIBUTE10" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESREFERENCE1" VARCHAR2(240) COLLATE "USING_NLS_COMP", "GLJELINESREFERENCE2" VARCHAR2(240) COLLATE "USING_NLS_COMP", "GLJELINESREFERENCE3" VARCHAR2(240) COLLATE "USING_NLS_COMP", "GLJELINESREFERENCE4" VARCHAR2(240) COLLATE "USING_NLS_COMP", "GLJELINESREFERENCE5" VARCHAR2(240) COLLATE "USING_NLS_COMP", "GLJELINESREFERENCE6" VARCHAR2(240) COLLATE "USING_NLS_COMP", "GLJELINESREFERENCE7" VARCHAR2(240) COLLATE "USING_NLS_COMP", "GLJELINESREFERENCE8" VARCHAR2(240) COLLATE "USING_NLS_COMP", "GLJELINESREFERENCE9" VARCHAR2(240) COLLATE "USING_NLS_COMP", "GLJELINESREFERENCE10" VARCHAR2(240) COLLATE "USING_NLS_COMP", "GLJELINESCURRENCYCONVERSIONDATE" DATE, "GLJELINESCURRENCYCONVERSIONRATE" NUMBER, "GLJELINESCURRENCYCONVERSIONTYPE" VARCHAR2(30) COLLATE "USING_NLS_COMP", "GLJELINESEFFECTIVEDATE" DATE, "GLJELINESIGNORERATEFLAG" VARCHAR2(1) COLLATE "USING_NLS_COMP", "GLJELINESSUBLEDGERDOCSEQUENCEID" NUMBER, "GLJELINESSUBLEDGERDOCSEQUENCEVALUE" NUMBER, "GLJELINESLASTUPDATEDBY" VARCHAR2(64) COLLATE "USING_NLS_COMP", "GLJELINESCREATEDBY" VARCHAR2(64) COLLATE "USING_NLS_COMP", "GLJELINESCREATIONDATE" TIMESTAMP (6), "GLJELINESATTRIBUTE11" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESATTRIBUTE12" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESATTRIBUTE13" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESATTRIBUTE14" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESATTRIBUTE15" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESATTRIBUTE16" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESATTRIBUTE17" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESATTRIBUTE18" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESATTRIBUTE19" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESATTRIBUTE20" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESATTRIBUTECATEGORY3" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESGLOBALATTRIBUTE1" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESGLOBALATTRIBUTE10" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESGLOBALATTRIBUTE11" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESGLOBALATTRIBUTE12" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESGLOBALATTRIBUTE13" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESGLOBALATTRIBUTE14" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESGLOBALATTRIBUTE15" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESGLOBALATTRIBUTE16" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESGLOBALATTRIBUTE17" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESGLOBALATTRIBUTE18" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESGLOBALATTRIBUTE19" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESGLOBALATTRIBUTE2" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESGLOBALATTRIBUTE20" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESGLOBALATTRIBUTE3" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESGLOBALATTRIBUTE4" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESGLOBALATTRIBUTE5" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESGLOBALATTRIBUTE6" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESGLOBALATTRIBUTE7" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESGLOBALATTRIBUTE8" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESGLOBALATTRIBUTE9" VARCHAR2(150) COLLATE "USING_NLS_COMP", "GLJELINESGLOBALATTRIBUTECATEGORY" VARCHAR2(30) COLLATE "USING_NLS_COMP", "GLJELINESGLOBALATTRIBUTEDATE1" DATE, "GLJELINESGLOBALATTRIBUTEDATE2" DATE, "GLJELINESGLOBALATTRIBUTEDATE3" DATE, "GLJELINESGLOBALATTRIBUTEDATE4" DATE, "GLJELINESGLOBALATTRIBUTEDATE5" DATE, "GLJELINESGLOBALATTRIBUTENUMBER1" NUMBER, "GLJELINESGLOBALATTRIBUTENUMBER2" NUMBER, "GLJELINESGLOBALATTRIBUTENUMBER3" NUMBER, "GLJELINESGLOBALATTRIBUTENUMBER4" NUMBER, "GLJELINESGLOBALATTRIBUTENUMBER5" NUMBER, "GLJELINESGLSLLINKID" NUMBER, "GLJELINESGLSLLINKTABLE" VARCHAR2(30) COLLATE "USING_NLS_COMP", "GLJELINESLASTUPDATELOGIN" VARCHAR2(32) COLLATE "USING_NLS_COMP", "GLJELINESOBJECTVERSIONNUMBER" NUMBER(9,0),

```
    "ACCT_PERIOD_NUM_V" NUMBER GENERATED ALWAYS AS (CASE SUBSTR(GLJELINESPERIODNAME,0,2)
        WHEN '13' THEN TO_NUMBER(TO_CHAR(TO_DATE(SUBSTR(GLJELINESPERIODNAME,4,LENGTH(GLJELINESPERIODNAME)), 'MON-YYYY'),'YYYY') || '13')
        ELSE  TO_NUMBER(TO_CHAR(TO_DATE(GLJELINESPERIODNAME, 'MON-YYYY'),'YYYYMM'))
        END) VIRTUAL)
;
```

Create GL_DW_JOURNALLINES with virtual column acting as partition key, Partitions and local indexes

```
CREATE TABLE "ATTFINCUST"."ATGL_DW_JOURNALLINES"
   (  "JEHEADERID" NUMBER(18,0),
  "JELINENUM" NUMBER(18,0),
  "GLJELINESLASTUPDATEDATE" TIMESTAMP (6),
  "GLJELINESPERIODNAME" VARCHAR2(15) COLLATE "USING_NLS_COMP",
  "GLJELINESDESCRIPTION" VARCHAR2(240) COLLATE "USING_NLS_COMP",
  "GLJELINESLEDGERID" NUMBER(18,0),
  "GLJELINESCURRENCYCODE" VARCHAR2(15) COLLATE "USING_NLS_COMP",
  "GLJELINESCODECOMBINATIONID" NUMBER(18,0),
  "GLJELINESSTATUS" VARCHAR2(1) COLLATE "USING_NLS_COMP",
  "GLJELINESENTEREDDR" NUMBER,
  "GLJELINESENTEREDCR" NUMBER,
  "GLJELINESACCOUNTEDDR" NUMBER,
  "GLJELINESACCOUNTEDCR" NUMBER,
  "GLJELINESSTATAMOUNT" NUMBER,
  "GLJELINESLINETYPECODE" VARCHAR2(20) COLLATE "USING_NLS_COMP",
  "GLJELINESATTRIBUTECATEGORY" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESATTRIBUTE1" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESATTRIBUTE2" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESATTRIBUTE3" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESATTRIBUTE4" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESATTRIBUTE5" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESATTRIBUTE6" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESATTRIBUTE7" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESATTRIBUTE8" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESATTRIBUTE9" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESATTRIBUTE10" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESREFERENCE1" VARCHAR2(240) COLLATE "USING_NLS_COMP",
  "GLJELINESREFERENCE2" VARCHAR2(240) COLLATE "USING_NLS_COMP",
  "GLJELINESREFERENCE3" VARCHAR2(240) COLLATE "USING_NLS_COMP",
  "GLJELINESREFERENCE4" VARCHAR2(240) COLLATE "USING_NLS_COMP",
  "GLJELINESREFERENCE5" VARCHAR2(240) COLLATE "USING_NLS_COMP",
  "GLJELINESREFERENCE6" VARCHAR2(240) COLLATE "USING_NLS_COMP",
  "GLJELINESREFERENCE7" VARCHAR2(240) COLLATE "USING_NLS_COMP",
  "GLJELINESREFERENCE8" VARCHAR2(240) COLLATE "USING_NLS_COMP",
  "GLJELINESREFERENCE9" VARCHAR2(240) COLLATE "USING_NLS_COMP",
  "GLJELINESREFERENCE10" VARCHAR2(240) COLLATE "USING_NLS_COMP",
  "GLJELINESCURRENCYCONVERSIONDATE" DATE,
  "GLJELINESCURRENCYCONVERSIONRATE" NUMBER,
  "GLJELINESCURRENCYCONVERSIONTYPE" VARCHAR2(30) COLLATE "USING_NLS_COMP",
  "GLJELINESEFFECTIVEDATE" DATE,
  "GLJELINESIGNORERATEFLAG" VARCHAR2(1) COLLATE "USING_NLS_COMP",
  "GLJELINESSUBLEDGERDOCSEQUENCEID" NUMBER,
  "GLJELINESSUBLEDGERDOCSEQUENCEVALUE" NUMBER,
  "GLJELINESLASTUPDATEDBY" VARCHAR2(64) COLLATE "USING_NLS_COMP",
  "GLJELINESCREATEDBY" VARCHAR2(64) COLLATE "USING_NLS_COMP",
  "GLJELINESCREATIONDATE" TIMESTAMP (6),
  "GLJELINESATTRIBUTE11" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESATTRIBUTE12" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESATTRIBUTE13" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESATTRIBUTE14" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESATTRIBUTE15" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESATTRIBUTE16" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESATTRIBUTE17" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESATTRIBUTE18" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESATTRIBUTE19" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESATTRIBUTE20" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESATTRIBUTECATEGORY3" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESGLOBALATTRIBUTE1" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESGLOBALATTRIBUTE10" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESGLOBALATTRIBUTE11" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESGLOBALATTRIBUTE12" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESGLOBALATTRIBUTE13" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESGLOBALATTRIBUTE14" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESGLOBALATTRIBUTE15" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESGLOBALATTRIBUTE16" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESGLOBALATTRIBUTE17" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESGLOBALATTRIBUTE18" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESGLOBALATTRIBUTE19" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESGLOBALATTRIBUTE2" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESGLOBALATTRIBUTE20" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESGLOBALATTRIBUTE3" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESGLOBALATTRIBUTE4" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESGLOBALATTRIBUTE5" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESGLOBALATTRIBUTE6" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESGLOBALATTRIBUTE7" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESGLOBALATTRIBUTE8" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESGLOBALATTRIBUTE9" VARCHAR2(150) COLLATE "USING_NLS_COMP",
  "GLJELINESGLOBALATTRIBUTECATEGORY" VARCHAR2(30) COLLATE "USING_NLS_COMP",
  "GLJELINESGLOBALATTRIBUTEDATE1" DATE,
  "GLJELINESGLOBALATTRIBUTEDATE2" DATE,
  "GLJELINESGLOBALATTRIBUTEDATE3" DATE,
  "GLJELINESGLOBALATTRIBUTEDATE4" DATE,
  "GLJELINESGLOBALATTRIBUTEDATE5" DATE,
  "GLJELINESGLOBALATTRIBUTENUMBER1" NUMBER,
  "GLJELINESGLOBALATTRIBUTENUMBER2" NUMBER,
  "GLJELINESGLOBALATTRIBUTENUMBER3" NUMBER,
  "GLJELINESGLOBALATTRIBUTENUMBER4" NUMBER,
  "GLJELINESGLOBALATTRIBUTENUMBER5" NUMBER,
  "GLJELINESGLSLLINKID" NUMBER,
  "GLJELINESGLSLLINKTABLE" VARCHAR2(30) COLLATE "USING_NLS_COMP",
  "GLJELINESLASTUPDATELOGIN" VARCHAR2(32) COLLATE "USING_NLS_COMP",
  "GLJELINESOBJECTVERSIONNUMBER" NUMBER(9,0),
    "ACCT_PERIOD_NUM_V" NUMBER GENERATED ALWAYS AS (CASE SUBSTR(GLJELINESPERIODNAME,0,2)
        WHEN '13' THEN TO_NUMBER(TO_CHAR(TO_DATE(SUBSTR(GLJELINESPERIODNAME,4,LENGTH(GLJELINESPERIODNAME)), 'MON-YYYY'),'YYYY') || '13')
        ELSE  TO_NUMBER(TO_CHAR(TO_DATE(GLJELINESPERIODNAME, 'MON-YYYY'),'YYYYMM'))
        END) VIRTUAL)

    PARTITION BY RANGE ("ACCT_PERIOD_NUM_V")
    (
        PARTITION "PART_PRE2018"  VALUES LESS THAN (201712) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_201801"  VALUES LESS THAN (201802) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_201802"  VALUES LESS THAN (201803) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_201803"  VALUES LESS THAN (201804) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_201804"  VALUES LESS THAN (201805) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_201805"  VALUES LESS THAN (201806) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_201806"  VALUES LESS THAN (201807) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_201807"  VALUES LESS THAN (201808) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_201808"  VALUES LESS THAN (201809) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_201809"  VALUES LESS THAN (201810) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_201810"  VALUES LESS THAN (201811) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_201811"  VALUES LESS THAN (201812) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_201812"  VALUES LESS THAN (201901) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_201901"  VALUES LESS THAN (201902) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_201902"  VALUES LESS THAN (201903) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_201903"  VALUES LESS THAN (201904) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_201904"  VALUES LESS THAN (201905) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_201905"  VALUES LESS THAN (201906) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_201906"  VALUES LESS THAN (201907) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_201907"  VALUES LESS THAN (201908) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_201908"  VALUES LESS THAN (201909) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_201909"  VALUES LESS THAN (201910) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_201910"  VALUES LESS THAN (201911) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_201911"  VALUES LESS THAN (201912) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_201912"  VALUES LESS THAN (202001) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202001"  VALUES LESS THAN (202002) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202002"  VALUES LESS THAN (202003) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202003"  VALUES LESS THAN (202004) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202004"  VALUES LESS THAN (202005) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202005"  VALUES LESS THAN (202006) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202006"  VALUES LESS THAN (202007) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202007"  VALUES LESS THAN (202008) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202008"  VALUES LESS THAN (202009) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202009"  VALUES LESS THAN (202010) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202010"  VALUES LESS THAN (202011) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202011"  VALUES LESS THAN (202012) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202012"  VALUES LESS THAN (202101) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202101"  VALUES LESS THAN (202102) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202102"  VALUES LESS THAN (202103) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202103"  VALUES LESS THAN (202104) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202104"  VALUES LESS THAN (202105) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202105"  VALUES LESS THAN (202106) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202106"  VALUES LESS THAN (202107) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202107"  VALUES LESS THAN (202108) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202108"  VALUES LESS THAN (202109) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202109"  VALUES LESS THAN (202110) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202110"  VALUES LESS THAN (202111) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202111"  VALUES LESS THAN (202112) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202112"  VALUES LESS THAN (202201) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202201"  VALUES LESS THAN (202202) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202202"  VALUES LESS THAN (202203) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202203"  VALUES LESS THAN (202204) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202204"  VALUES LESS THAN (202205) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202205"  VALUES LESS THAN (202206) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202206"  VALUES LESS THAN (202207) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202207"  VALUES LESS THAN (202208) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202208"  VALUES LESS THAN (202209) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202209"  VALUES LESS THAN (202210) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202210"  VALUES LESS THAN (202211) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202211"  VALUES LESS THAN (202212) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202212"  VALUES LESS THAN (202301) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202301"  VALUES LESS THAN (202302) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202302"  VALUES LESS THAN (202303) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202303"  VALUES LESS THAN (202304) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202304"  VALUES LESS THAN (202305) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202305"  VALUES LESS THAN (202306) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202306"  VALUES LESS THAN (202307) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202307"  VALUES LESS THAN (202308) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202308"  VALUES LESS THAN (202309) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202309"  VALUES LESS THAN (202310) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202310"  VALUES LESS THAN (202311) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202311"  VALUES LESS THAN (202312) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA" ,
        PARTITION "PART_202312"  VALUES LESS THAN (202401) SEGMENT CREATION IMMEDIATE
        TABLESPACE "DATA"
    )
;


ALTER TABLE "GL_DW_JOURNALLINES"
ADD CONSTRAINT "ATGL_DW_JOURNALLINE_PK1"
PRIMARY KEY ("JEHEADERID", "JELINENUM","ACCT_PERIOD_NUM_V")
USING INDEX LOCAL
PCTFREE 10 INITRANS 20 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "DATA_2"  ENABLE

;



Partition column
"ACCT_PERIOD_NUM_V" NUMBER GENERATED ALWAYS AS (
CASE SUBSTR(GLJELINESPERIODNAME,0,2)
    WHEN '13' THEN TO_NUMBER(TO_CHAR(TO_DATE(SUBSTR(GLJELINESPERIODNAME,4,LENGTH(GLJELINESPERIODNAME)), 'MON-YYYY'),'YYYY') || '13')
    ELSE  TO_NUMBER(TO_CHAR(TO_DATE(GLJELINESPERIODNAME, 'MON-YYYY'),'YYYYMM'))END) VIRTUAL)
```

Extraction / Load Strategy from Oracle Data Integrator

To extract data from Fusion with ODI using BICC, the recommended Loading Knowledge Module is *LKM BICC to ADW External Table*.

It is designed for the purpose of extracting data from Oracle Business Intelligence Cloud (BICC) and loading it into Oracle Autonomous Data Warehouse (ADW) External Tables. This knowledge module serves several key purposes:

1. **Data Extraction:** The primary purpose of the LKM "BICC to ADW External Table" is to extract data from BICC, which is typically used for business intelligence and reporting, and prepare it for loading into ADW External Tables. BICC contains a wealth of structured data that may be valuable for various analytical purposes.
2. **Data Transformation:** The LKM is responsible for transforming the data as needed to match the structure and requirements of the target ADW External Tables. This transformation may include data type conversions, data cleansing, and data mapping.
3. **Data Loading:** The LKM orchestrates the process of loading the transformed data into ADW External Tables. It ensures that the data is correctly and efficiently loaded into the target tables.
4. **Data Integration:** By using this knowledge module, organizations can integrate their BICC data with other data sources stored in ADW External Tables, enabling a consolidated view of data for reporting, analysis, and business intelligence.
5. **Incremental Loading:** The LKM can be configured to support incremental data loading. This means it can identify and load only the data that has changed or been added since the last load, improving efficiency and performance.
6. **Automation:** By using an appropriate LKM like "BICC to ADW External Table," organizations can automate the data extraction, transformation, and loading processes, reducing manual effort and potential errors.
7. **Optimized for ADW External Tables:** This knowledge module is optimized for working with ADW External Tables, which are designed for handling and querying large volumes of structured data efficiently. ADW External Tables can be queried using SQL just like native database tables.
8. **Seamless Integration:** The LKM ensures seamless integration between the BICC data and ADW External Tables, allowing organizations to leverage the data stored in BICC for advanced analytics and reporting within their data warehouse environment.
9. **Data Synchronization:** By using this LKM, organizations can ensure that the data in ADW External Tables is kept in sync with the data in BICC, enabling real-time or near-real-time analytics.

This LKM has an option named "LAST_LOAD_DATE" option that is used to specify a timestamp value that helps identify the last time data was loaded from Oracle Business Intelligence Cloud (BICC) into Oracle Autonomous Data Warehouse (ADW) External Tables. This option is typically used for incremental data loading and synchronization between these two systems. Here's the purpose of the "LAST_LOAD_DATE" option:

1. **Incremental Data Loading:** The "LAST_LOAD_DATE" option is particularly valuable when you want to perform incremental data loading. Incremental loading involves extracting and loading only the data that has changed or been added since the last load. This approach is more efficient than a full data load when dealing with large datasets.
2. **Timestamp-Based Tracking:** The "LAST_LOAD_DATE" option is used to track the timestamp of the most recent data extraction from BICC. This timestamp is typically saved in a control table or configuration file for reference.
3. **Comparison with Source Data:** During subsequent data extraction and loading processes, ODI compares the "LAST_LOAD_DATE" with the timestamps in the source data from BICC. It identifies records that have been updated or inserted since the last load by comparing the timestamps.

4. **Filtering New and Updated Data:** By comparing timestamps, ODI can filter out data that has not changed since the last load. This means only new or updated records need to be extracted and loaded into ADW External Tables.
5. **Efficiency and Performance:** The use of "LAST_LOAD_DATE" for incremental loading improves the efficiency and performance of the ETL (Extract, Transform, Load) process. It reduces the volume of data that needs to be transferred and loaded, saving time and resources.
6. **Ensuring Data Consistency:** Incremental loading with a "LAST_LOAD_DATE" approach helps ensure that ADW External Tables remain synchronized with the latest data available in BICC, avoiding the need for a complete data refresh.

To use the "LAST_LOAD_DATE" option effectively in the LKM "BICC to ADW External Table," you would typically set it to the timestamp of the last successful data extraction from BICC. This value should be updated after each successful ETL process to reflect the most recent data load date. By maintaining and using this timestamp, you can achieve efficient and accurate incremental data loading from BICC to ADW External Tables.

Integration Strategy

The recommended integration strategy is to use the Integration Knowledge Module *IKM Oracle Merge*.

The IKM (Integration Knowledge Module) "Oracle Merge" in Oracle Data Integrator (ODI) is designed for the purpose of performing Merge operations, also known as "upsert" operations, when loading data into target tables in an Oracle database. The primary purpose of the IKM Oracle Merge is to efficiently synchronize and merge data from a source to a target table in the database. Here are the main purposes and features of the IKM Oracle Merge in ODI:

1. **Upsert Operations:** The primary function of the IKM Oracle Merge is to perform upsert operations, which combine "insert" and "update" operations into a single, efficient operation. It ensures that data is either inserted into the target table if it doesn't exist or updated if it does, based on a specified condition.
2. **Efficient Data Synchronization:** The IKM Oracle Merge is particularly useful for synchronizing data between source and target tables, especially when dealing with changing data that requires periodic updates.
3. **Eliminating Data Redundancy:** By using the Merge statement, the IKM Oracle Merge helps eliminate data redundancy, ensuring that the target table contains only unique records and that data integrity is maintained.
4. **Reduced Database Load:** The Merge operation is typically more efficient than separate insert and update operations because it minimizes the number of SQL statements executed and reduces the load on the database.
5. **Conditional Logic:** The IKM Oracle Merge allows you to define conditional logic to determine how the upsert operation is performed. You specify the criteria for identifying which records should be inserted and which should be updated.
6. **Automatic Handling of Primary Keys:** The IKM Oracle Merge handles the primary key and unique constraint checks automatically, ensuring that data integrity is maintained while performing upserts.
7. **Logging and Error Handling:** The knowledge module typically includes logging and error handling mechanisms to capture and manage any issues that may arise during the upsert process.
8. **Performance Optimization:** The IKM Oracle Merge is optimized for performance, making it a suitable choice for efficiently handling large datasets and minimizing database processing time.
9. **Integration with ODI Mappings:** The IKM Oracle Merge can be used in ODI mappings to define the data flow and transformations from source to target, including the Merge operation.
10. **Customization:** While the IKM Oracle Merge provides a standard mechanism for performing Merge operations, it can be customized to fit specific business requirements or to implement complex merge logic.

This IKM has an option to specify HINTS on merge statement. The below hints have proven to be executing the merge operation faster in some cases. They should be tested, benchmarked and modified based on execution results.

```
MERGE /*+ append */
MERGE /*+ append parallel (4) */
MERGE /*+ append leading(GL_DW_JOURNALLINES) use_nl(PS MERGE_SUBQUERY) parallel (4) */
```

When the volume of data for full/initial or incremental is high, the recommendation is to use a partition exchange approach. The IKM of choice in the case is IKM Oracle Partition Exchange

The IKM (Integration Knowledge Module) Oracle Partition Exchange in Oracle Data Integrator (ODI) serves the purpose of automating the management of partitions within an Oracle database. Specifically, it is designed for performing partition exchange operations, which allow for the efficient movement of data between tables and partitions.

Here are the main purposes and features of the IKM Oracle Partition Exchange in ODI:

1. **Partition Management:** The primary purpose of the IKM Oracle Partition Exchange is to manage partitions within an Oracle database. It automates the process of exchanging data between two tables, one of which may be a partitioned table and the other an empty staging table.
2. **Data Archiving and Retention:** The IKM Oracle Partition Exchange is often used for archiving or data retention purposes. It allows you to move older data from a partitioned table into a separate archive table, which can be useful for historical data preservation and compliance.
3. **Performance Optimization:** Partition exchange is a highly efficient operation, especially for data archiving or purging scenarios. It minimizes the need to physically copy data and can be significantly faster than traditional data movement operations.
4. **Data Maintenance:** The IKM Oracle Partition Exchange facilitates the maintenance of partitioned tables by allowing you to remove data that is no longer needed in the main table and move it to an archive or backup table.
5. **Parallel Processing:** This knowledge module often leverages parallel processing capabilities of the Oracle database, making partition exchanges faster and more efficient.
6. **Reduced Impact on Transaction Log:** Partition exchange operations typically have a reduced impact on the transaction log because they involve metadata changes rather than data copying.
7. **Data Lifecycle Management:** The IKM Oracle Partition Exchange supports data lifecycle management, allowing you to automate the process of moving data to different storage locations or archive tables based on criteria such as age or other business rules.
8. **Customization:** While the IKM Oracle Partition Exchange provides a standard mechanism for performing partition exchange, it can be customized to fit specific business requirements or to implement complex partition exchange logic.

The A-Team has written two details blogs on how to use partition exchange approach. Please refer to the links below for detailed implementation guidance and also to get a better understanding of the following sections:

Using Oracle Partition Exchange with ODI

Configuring ODI with Oracle Partition Exchange

Initial Load Approach with Partition Exchange

The figure below shows how the IKM Oracle Partition Exchange Load performs the partition exchange operation for the table used for the example, called GL_DW_JOURNALLINE. This fact table has been partitioned by GL Period Name.

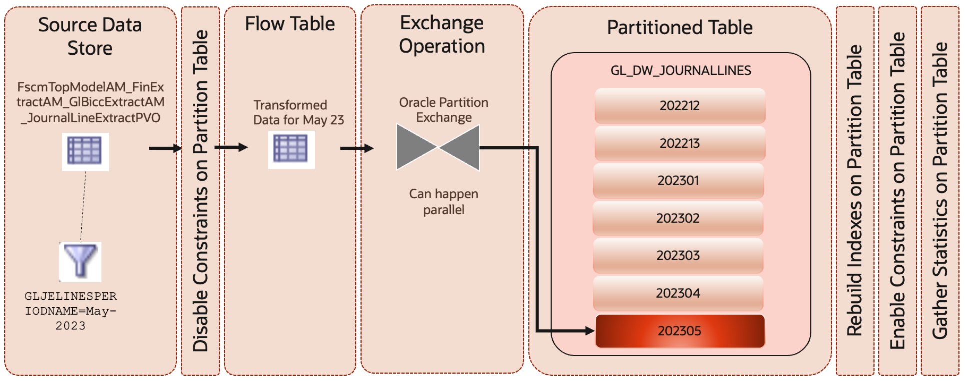

Source Data Store
FscmTopModelAM_FinExtractAM_GlBiccExtractAM_JournalLineExtractPVO
GLJELINESPERIODNAME=May-2023
Disable Constraints on Partition Table
Flow Table
Transformed Data for May 23
Exchange Operation
Oracle Partition Exchange
Can happen parallel
Partitioned Table
GL_DW_JOURNALLINES
202212
202213
202301
202302
202303
202304
202305
Rebuild Indexes on Partition Table
Enable Constraints on Partition Table
Gather Statistics on Partition Table

Figure 5: Partition Exchange with ODI

In the above example, the source data store called *FscmTopModelAM_FinExtractAM_GlBiccExtractAM_JournalLineExtractPVO* is used and a filter is applied to select data for the month of May 2023. At a minimum, the knowledge module performs the following three tasks:

- Applies the data filter, and transforms the source data.
- Loads the transformed data into the ODI flow table.
- Exchanges partition *202305* with the ODI flow table.

Some of the knowledge module tasks – such as disabling and enabling constraints, rebuilding indexes, and collecting statistics – are optional. These optional tasks can be controlled by the user via the knowledge module options. The next section of this article presents an overview of the KM tasks.

The figure below describes and implementation of the partition exchange approach in an ODI Package.

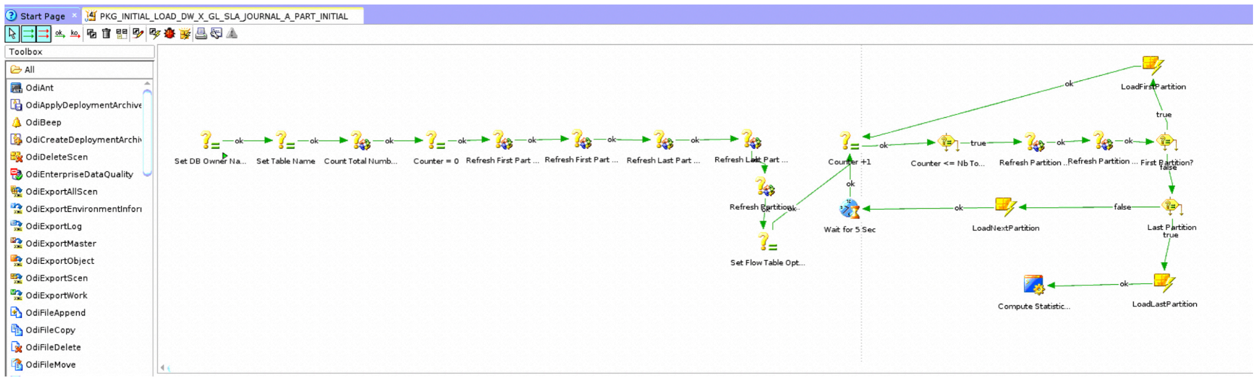


Figure 6: Partition Exchange Implementation in a package in ODI

The first steps are variables required by the process such as the schema/table name for the target load, the number of partition, the name of the first and last partitions as they have specific treatments, loop counters.

Then there is the loop that “loops” through each partition and execute the partition data load, with again specific treatment for the first and the last partitions.

For more details, please refer to the blogs below

Using Oracle Partition Exchange with ODI

Configuring ODI with Oracle Partition Exchange

Incremental Load

Once the data is loaded for the full / initial load, the incremental extract and load process can happen. The figure below describes this incremental process

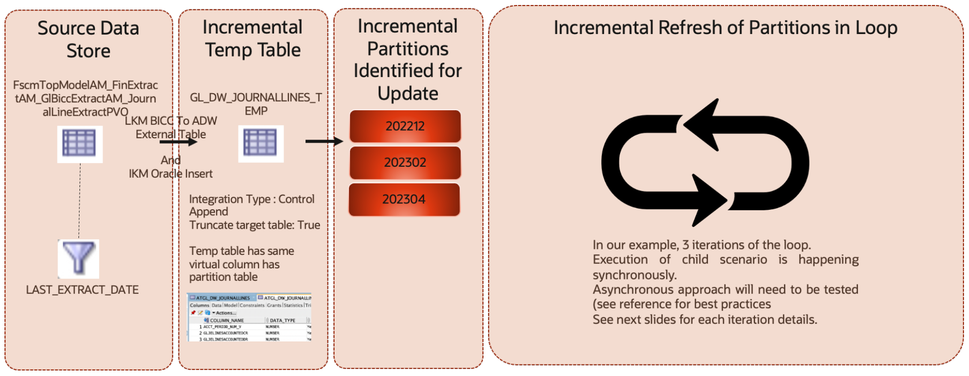

Source Data Store
FscmTopModelAM_FinExtractAM_GlBiccExtractAM_JournalLineExtractPVO
LAST_EXTRACT_DATE
LKM BICC To ADW External Table
And
IKM Oracle Insert
Incremental Temp Table
GL_DW_JOURNALLINES_TEMP
Integration Type : Control Append
Truncate target table: True
Temp table has same virtual column has partition table
ATGL_DW_JOURNALLINES
Columns
COLUMN_NAME
DATA_TYPE
ACCT_PERIOD_NUM_V
GLJELINESACCOUNTEDCR
GLJELINESACCOUNTEDDR
Incremental Partitions Identified for Update
202212
202302
202304
Incremental Refresh of Partitions in Loop
In our example, 3 iterations of the loop.
Execution of child scenario is happening synchronously.
Asynchronous approach will need to be tested (see reference for best practices
See next slides for each iteration details.

Figure 7: Partition Exchange for incremental load in ODI

The flow for incremental is as follow:

1. **Incremental Extract:** The BICC PVO is extracted incrementally, using the last extract date with the LKM BICC to ADW External Table.
2. **Load into the Temporary Table:** The BICC PVO extract is then loaded into a temporary table that has the same structure and virtual columns as the target partitioned table. The ODI IKM use in this step is the SQL Control Append and the temporary table is truncated before each load.
3. **Identification of the Partitions to Load:** Once the temporary table is loaded, a series of ODI artifacts, namely variables, are used to identify and store the list of partitions to be loaded
4. **Process Partition Exchanges in a Loop:** Each partition are then processed in a similar way as the full load.

In the example, there are three partitions to be loaded in the loop. In this example, the Flow Control option is used to validate the new dataset against the constraints of the partitioned table. The data validation is performed by the Oracle check knowledge module (CKM Oracle). If invalid records are found in the flow table, they are copied into an error table and removed from the flow table before the exchange operation.

Local indexes are included during the partition exchange operation. This is done by using the INCLUDING INDEXES value in the Partition Exchange Options. Thus, there is no need to rebuild local indexes after the exchange operation, since they will stay in a usable state after the exchange operation.

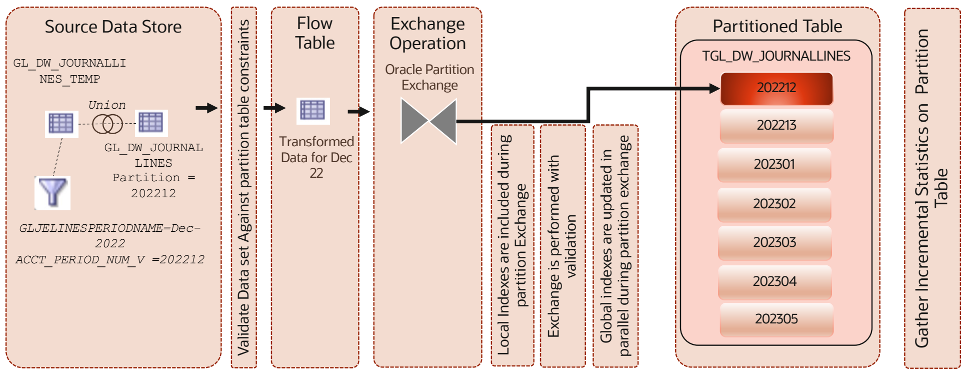
The exchange operation is performed with validation. This is done by using the WITH VALIDATION value in the Partition Exchange Options.

The KM does not disable or enable the table constraints for the partitioned table during the partition exchange operation. However, in the database, the table constraints for the partitioned table are enabled.

Global indexes are updated – in parallel – during the partition exchange operation. This is done by using the UPDATE INDEXES PARALLEL value in the Partition Exchange Options. Thus, global indexes will stay usable during the exchange operation.

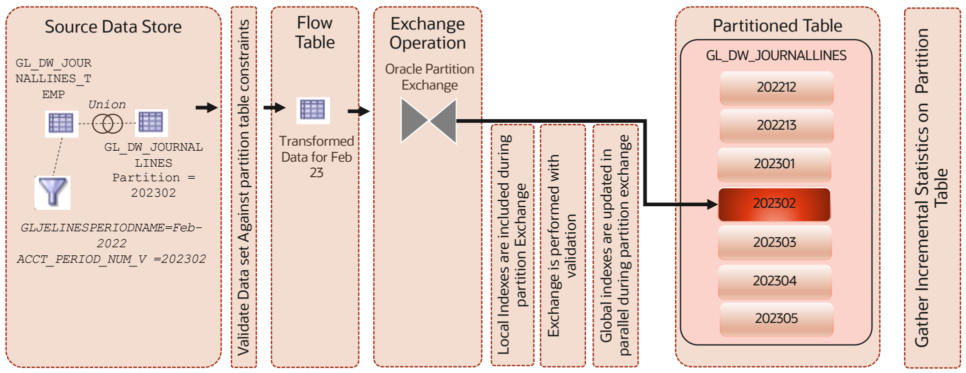
Incremental statistics will be gathered after the partition exchange operation. The statistic and publish preferences of the partitioned table have been set to ***incremental*** during the initial upload operation. Thus, table statistics are gathered incrementally. If the partitioned table is composite, both subpartition-level and partition-level statistics are gathered for the exchanged partition. If the partitioned table is single-level, only partition-level statistics are gathered for the exchanged partition.

1. **First iteration of the loop :**



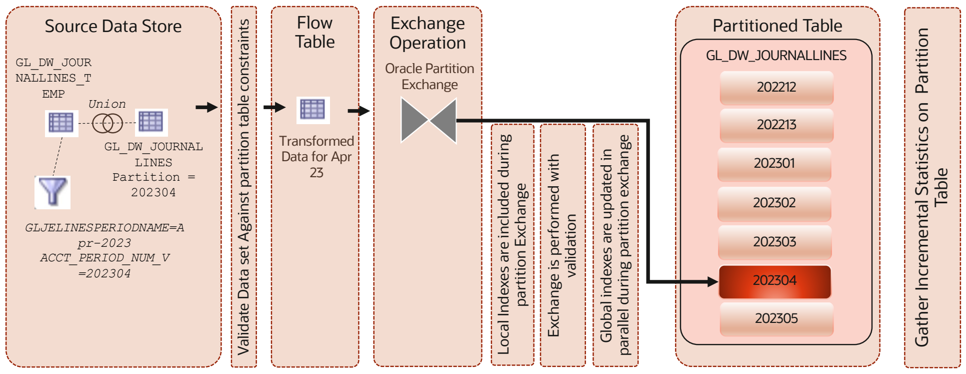
Source Data Store
GL_DW_JOURNALLINES_TEMP
Union
GL_DW_JOURNAL LINES Partition = 202212
GLJELINESPERIODNAME=Dec-2022
ACCT_PERIOD_NUM_V =202212
Validate Data set Against partition table constraints
Flow Table
Transformed Data for Dec 22
Exchange Operation
Oracle Partition Exchange
Local Indexes are included during partition Exchange
Exchange is performed with validation
Global indexes are updated in parallel during partition exchange
Partitioned Table
TGL_DW_JOURNALLINES
202212
202213
202301
202302
202303
202304
202305
Gather Incremental Statistics on Partition Table

2. **First iteration of the loop :**



Source Data Store
GL_DW_JOURNALLINES_TEMP
Union
GL_DW_JOURNAL LINES Partition = 202302
GLJELINESPERIODNAME=Feb-2022
ACCT_PERIOD_NUM_V =202302
Validate Data set Against partition table constraints
Flow Table
Transformed Data for Feb 23
Exchange Operation
Oracle Partition Exchange
Local Indexes are included during partition Exchange
Exchange is performed with validation
Global indexes are updated in parallel during partition exchange
Partitioned Table
GL_DW_JOURNALLINES
202212
202213
202301
202302
202303
202304
202305
Gather Incremental Statistics on Partition Table

3. **First iteration of the loop :**



Source Data Store
GL_DW_JOURNALLINES_TEMP
Union
GL_DW_JOURNAL LINES Partition = 202304
GLJELINESPERIODNAME=Apr-2023
ACCT_PERIOD_NUM_V =202304
Validate Data set Against partition table constraints
Flow Table
Transformed Data for Apr 23
Exchange Operation
Oracle Partition Exchange
Local Indexes are included during partition Exchange
Exchange is performed with validation
Global indexes are updated in parallel during partition exchange
Partitioned Table
GL_DW_JOURNALLINES
202212
202213
202301
202302
202303
202304
202305
Gather Incremental Statistics on Partition Table

Soft Deletes

In a data warehouse context, "soft deletes" refer to a data management technique where records are not physically removed from the data warehouse when they are no longer needed or are marked for deletion. Instead, soft deletes involve the use of indicators or flags to mark records as deleted without actually deleting them from the database.

This technique is employed for several reasons:

1. **Data Retention and History:** Soft deletes allow organizations to maintain a historical record of changes to the data. Even if a record is no longer considered active, the soft-deleted version remains in the data warehouse, preserving the history of the data over time.
2. **Data Recovery:** By not physically deleting records, organizations have the flexibility to recover or restore soft-deleted data when needed. This can be important for auditing, compliance, or data recovery purposes.
3. **Data Integrity:** Soft deletes help maintain data integrity by keeping the referential integrity of the database intact. Records that are marked as deleted can still be used in relationships or data links without disrupting the structure of the data warehouse.
4. **Efficient Data Management:** Soft deletes can be more efficient than physical deletion, especially in large data warehouses. Physically deleting records can be resource-intensive and time-consuming, whereas soft deletes are typically faster and less resource-intensive.
5. **Auditing and Compliance:** Soft deletes can assist in meeting regulatory and compliance requirements, as they provide a clear audit trail of data changes and deletions. Organizations can demonstrate that data was not lost or deleted without proper authorization.
6. **Data Archiving:** Soft deletes can be part of a data archiving strategy, where historical or less frequently accessed data is moved to an archive table or storage to free up space in the main data warehouse while preserving the data for compliance or reporting.
7. **Data Reversion:** Soft deletes enable users to revert to a previous state of data when necessary. This is valuable in scenarios where data changes have caused issues, and a rollback to a previous state is needed.
8. **Data Consistency:** When using soft deletes, data consistency is maintained because records can still be referenced even if they are marked as deleted. This is especially important in data warehousing, where data relationships are crucial for analytics.
9. **User Error Handling:** Soft deletes provide a safety net for users who may accidentally delete important data. In cases of accidental deletion, the data can be recovered more easily.

In practical terms, soft deletes are implemented by adding a "deleted" flag or similar indicator to the data records. When querying the data, application logic or SQL queries can filter out records with the "deleted" flag to ensure that only active data is presented to users. Soft delete indicators can be used for various data elements, such as rows in tables, columns within rows, or even entire tables.

Overall, soft deletes in a data warehouse context provide a balance between efficient data management, historical data retention, and data integrity, making them a valuable strategy for organizations with complex data needs.

Primary key extracts in the context of Oracle Business Intelligence Cloud (BICC) serve several important purposes related to data extraction and integration. The primary key extracts are often used when setting up ETL (Extract, Transform, Load) processes for moving data from BICC to other systems or data warehouses. Here are the key purposes of primary key extracts in BICC:

- **Data Integration:** Primary key extracts are used to identify and extract data records from BICC, ensuring that each record is uniquely identified by its primary key. This is crucial for data integration, as it allows you to match and synchronize records between BICC and other systems.
- **Data Uniqueness:** Primary keys ensure the uniqueness of each record in a database or data source. When extracting data from BICC, primary key extracts ensure that no duplicate records are transferred to the target system, maintaining data integrity.
- **Data Quality:** By using primary keys, data quality is improved because records can be accurately matched and compared between source and target systems. This helps in identifying and resolving data inconsistencies and errors.
- **Incremental Data Extraction:** Primary key extracts are essential for incremental data extraction. By tracking the primary key values of previously extracted records, you can extract only new or changed data during subsequent ETL processes, which is more efficient than re-extracting all data.
- **Data Synchronization:** Primary key extracts are used to synchronize data between systems. This is especially important when data in BICC needs to be synchronized with a data warehouse or other databases, ensuring that changes in one system are reflected in the other.
- **Referential Integrity:** In cases where BICC data is related to other data sources through foreign keys, primary key extracts enable the maintenance of referential integrity when integrating data across systems.

If the record count of a PVO is low and its full extract time is acceptable (typically under five minutes), then the best way to handle data integrity is to perform a full data extract from BICC.

In the contrary, the option of extracting primary keys from Fusion SaaS should be the route to handle records that have been deleted in Fusion SaaS and propagate the deletion information to the target table.

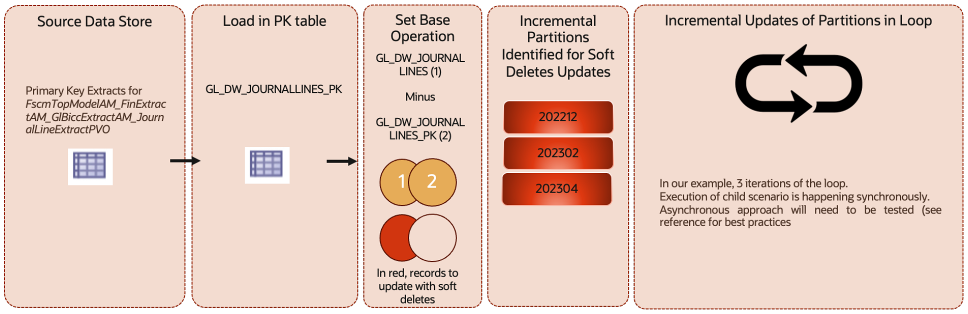

Source Data Store
Primary Key Extracts for *FscmTopModelAM_FinExtractAM_GlBiccExtractAM_JournalLineExtractPVO*
Load in PK table
GL_DW_JOURNALLINES_PK
Set Base Operation
GL_DW_JOURNAL LINES (1)
Minus
GL_DW_JOURNAL LINES_PK (2)
1
2
In red, records to update with soft deletes
Incremental Partitions Identified for Soft Deletes Updates
202212
202302
202304
Incremental Updates of Partitions in Loop
In our example, 3 iterations of the loop.
Execution of child scenario is happening synchronously.
Asynchronous approach will need to be tested (see reference for best practices

Figure 8: Partition Exchange for soft deletes updates in ODI

Conclusion

This document discussed the importance of up-to-date data in various business domains, including Finance, Supply Chain Management, and Customer Experience. It emphasizes that near real-time data is crucial for informed decision-making, operational efficiency, regulatory compliance, and customer satisfaction.

It also introduced a technical architectural pattern for extracting Oracle Fusion Application Cloud data at a higher frequency, outlining the benefits and use cases for each functional domain.

It covered aspects like accurate financial reporting, improved supply chain visibility, and enhanced customer support. Additionally, it described a technical solution architecture involving Oracle Cloud Infrastructure (OCI) for efficiently extracting and managing data from Oracle Fusion SaaS Applications, highlighting components like Oracle Data Integrator, Data Extraction, Data Storage, Data Transformation, Load, and Analytics.

In summary, the text underscored the significance of up to date real-time data in modern business operations and presents a technical solution architecture for achieving high-frequency data extraction and management in Oracle Fusion SaaS Applications on OCI.

References

A-Team Blogs on Partition Exchange

Using Oracle Partition Exchange with ODI

Configuring ODI with Oracle Partition Exchange

BI Cloud Connector

BI Cloud Connector Performance Recommendations

A-Team BICC Blogs

Database Design Best Practices

Statistics gather best practices

Get the best out of partitioning

Managing Automatic Partitioning on Autonomous Database Oracle Documentation

Automatic Partitioning with Autonomous Database Blog